The Prompt Report: A Systematic Survey of Prompt Engineering Techniques

Sander Schulhoff[1,2*] **Michael Ilie**[1*] **Nishant Balepur**[1] **Konstantine Kahadze**[1]
Amanda Liu[1] **Chenglei Si**[4] **Yinheng Li**[5] **Aayush Gupta**[1] **HyoJung Han**[1] **Sevien Schulhoff**[1]
Pranav Sandeep Dulepet[1] **Saurav Vidyadhara**[1] **Dayeon Ki**[1] **Sweta Agrawal**[12] **Chau Pham**[13]
Gerson Kroiz **Feileen Li**[1] **Hudson Tao**[1] **Ashay Srivastava**[1] **Hevander Da Costa**[1] **Saloni Gupta**[1]
Megan L. Rogers[8] **Inna Goncearenco**[9] **Giuseppe Sarli**[9,10] **Igor Galynker**[11]
Denis Peskoff[7] **Marine Carpuat**[1] **Jules White**[6] **Shyamal Anadkat**[3] **Alexander Hoyle**[1] **Philip Resnik**[1]
[1] University of Maryland [2] Learn Prompting [3] OpenAI [4] Stanford [5] Microsoft [6] Vanderbilt [7] Princeton
[8] Texas State University [9] Icahn School of Medicine [10] ASST Brianza
[11] Mount Sinai Beth Israel [12] Instituto de Telecomunicações [13] University of Massachusetts Amherst
sschulho@umd.edu milie@umd.edu resnik@umd.edu

Abstract

Generative Artificial Intelligence (GenAI) systems are increasingly being deployed across diverse industries and research domains. Developers and end-users interact with these systems through the use of prompting and prompt engineering. Although prompt engineering is a widely adopted and extensively researched area, it suffers from conflicting terminology and a fragmented ontological understanding of what constitutes an effective prompt due to its relatively recent emergence. We establish a structured understanding of prompt engineering by assembling a taxonomy of prompting techniques and analyzing their applications. We present a detailed vocabulary of 33 vocabulary terms, a taxonomy of 58 LLM prompting techniques, and 40 techniques for other modalities. Additionally, we provide best practices and guidelines for prompt engineering, including advice for prompting engineering ChatGPT and other state-of-the-art (SOTA) LLMs. We further present a meta-analysis of the entire literature on natural language prefix-prompting. As a culmination of these efforts, this paper presents the most comprehensive survey on prompt engineering to date.

Contents

1 Introduction

Transformer-based LLMs are widely deployed in consumer-facing, internal, and research settings (Bommasani et al., 2021). Typically, these models rely on the user providing an input "prompt" to which the model produces an output in response. Such prompts may be textual—"Write a poem about trees."—or take other forms: images, audio, videos, or a combination thereof. The ability to prompt models, particularly prompting with natural language, makes them easy to interact with and use flexibly across a wide range of use cases.

Knowing how to effectively structure, evaluate, and perform other tasks with prompts is essential to using these models. Empirically, better prompts lead to improved results across a wide range of tasks (Wei et al., 2022b; Liu et al., 2023b; Schulhoff, 2022). A large body of literature has grown around the use of prompting to improve results and the number of prompting techniques is rapidly increasing.

However, as prompting is an emerging field, the use of prompts continues to be poorly understood, with only a fraction of existing terminologies and techniques being well-known among practitioners. We perform a large-scale review of prompting techniques to create a robust resource of terminology and techniques in the field. We expect this to be the first iteration of terminologies that will develop over time. We maintain an up-to-date list of terms and techniques at LearnPrompting.org.

Scope of Study We create a broad directory of prompting techniques, that can be quickly understood and easily implemented for rapid experimentation by developers and researchers. To this end, we limit our study to focus on prefix prompts (Shin et al., 2020a) rather than cloze prompts (Petroni et al., 2019; Cui et al., 2021), because modern LLM transformer architectures widely employ prefix prompts and provide robust support for both developers and researchers (Brown et al., 2020; Google, 2023; Touvron et al., 2023). Additionally, we refined our focus to hard (discrete) prompts rather than soft (continuous) prompts and leave out papers that make use of techniques using gradient-based updates (i.e. fine-tuning). Hard prompts contain only tokens (vectors) that correspond to words in the model's vocabulary, while soft prompts may contain tokens that have no corresponding word in the vocabulary.

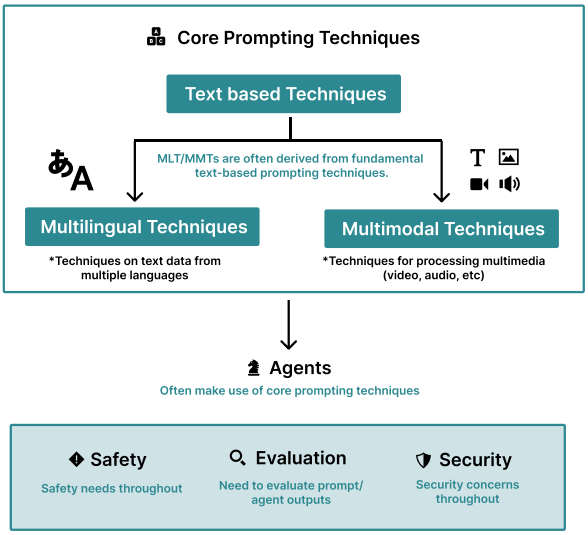

Figure 1.1: Categories within the field of prompting are interconnected. We discuss 7 core categories that are well described by the papers within our scope.

Finally, we only study task-agnostic techniques. These decisions keep the work approachable to less technical readers and maintain a manageable scope.

Sections Overview We conducted a machine-assisted systematic review grounded in the PRISMA process (Page et al., 2021) (Section 2.1) to identify 58 different text-based prompting techniques, from which we create a taxonomy with a robust terminology of prompting terms (Section 1.2).

Our goal is to provide a roadmap for the community when considering which prompting techniques to use (Figure 1.1). While much literature on prompting focuses on English-only settings, we also discuss multilingual techniques (Section 3.1). Given the rapid growth in multimodal prompting, where prompts may include media such as images, we also expand our scope to multimodal techniques (Section 3.2). Many multilingual and multimodal prompting techniques are direct extensions of English text-only prompting techniques.

As prompting techniques grow more complex, they have begun to incorporate external tools, such as Internet browsing and calculators. We use the

term "agents" to describe these types of prompting techniques (Section 4.1).

It is important to understand how to evaluate the outputs of agents and prompting techniques to ensure accuracy and avoid hallucinations. Thus, we discuss ways of evaluating these outputs (Section 4.2). We also discuss security (Section 5.1) and safety measures (Section 5.2) for designing prompts that reduce the risk of harm to companies and users.

Finally, we apply prompting techniques in two case studies (Section 6.1). In the first, we test a range of prompting techniques against the commonly used benchmark MMLU (Hendrycks et al., 2021). In the second, we explore in detail an example of manual prompt engineering on a significant, real-world use case, identifying signals of frantic hopelessness–a top indicator of suicidal crisis–in the text of individuals seeking support (Schuck et al., 2019a). We conclude with a discussion of the nature of prompting and its recent development (Section 8).

1.1 What is a Prompt?

A prompt is an input to a Generative AI model, that is used to guide its output (Meskó, 2023; White et al., 2023; Heston and Khun, 2023; Hadi et al., 2023; Brown et al., 2020). Prompts may consist of text, image, sound, or other media. Some examples of prompts include the text, “write a three paragraph email for a marketing campaign for an accounting firm”, a photograph of a piece of paper with the words “what is 10*179” written on it, or a recording of an online meeting, with the instructions “summarize this”. Prompts usually have some text component, but this may change as non-text modalities become more common.

Prompt Template Prompts are often constructed via a prompt template (Shin et al., 2020b). A prompt template is a function that contains one or more variables which will be replaced by some media (usually text) to create a prompt. This prompt can then be considered to be an *instance* of the template.

Consider applying prompting to the task of binary classification of tweets. Here is an initial prompt template that can be used to classify inputs.

> Classify the tweet as positive or negative:
> {TWEET}

> Write a poem about trees.

> Write a poem about the following topic:
> {USER_INPUT}

Figure 1.2: Prompts and prompt templates are distinct concepts; a prompt template becomes a prompt when input is inserted into it.

Each tweet in the dataset would be inserted into a separate instance of the template and the resulting prompt would be given to a LLM for inference.

1.2 Terminology

1.2.1 Components of a Prompt

There are a variety of common components included in a prompt. We summarize the most commonly used components and discuss how they fit into prompts (Figure 1.3).

Directive Many prompts issue a directive in the form of an instruction or question.[1] This is the core intent of the prompt, sometimes simply called the "intent". For example, here is an instance of a prompt with a single instruction:

> Tell me five good books to read.

Directives can also be implicit, as in this one-shot case, where the directive is to perform English to Spanish translation:

> Night: Noche
> Morning:

Examples Examples, also known as exemplars or shots, act as demonstrations that guide the GenAI to accomplish a task. The above prompt is a One-Shot (i.e. one example) prompt.

Output Formatting It is often desirable for the GenAI to output information in certain formats, for example, CSV, Markdown, XML, or even custom formats(Xia et al., 2024). Structuring outputs may reduce performance on some tasks (Tam et al., 2024). However, Kurt (2024) point out various

[1] “Directives”, from Searle (1969), are a type of speech act intended to encourage an action, and have been invoked in models of human-computer dialogue Morelli et al. (1991).

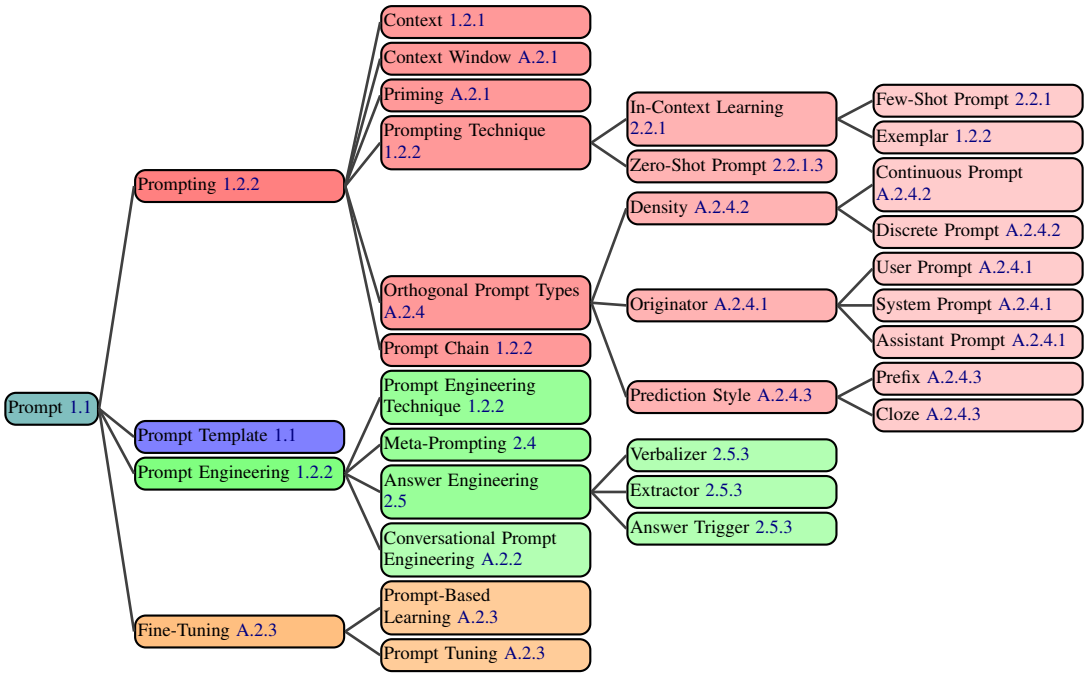

Figure 1.3: A Terminology of prompting. Terms with links to the appendix are not sufficiently critical to describe in the main paper, but are important to the field of prompting. Prompting techniques are shown in Figure 2.2

.

flaws in Tam et al. (2024) and show that structuring outputs may actually improve performance. Here is an example of how you might format a prompt to output information as a CSV:

> {PARAGRAPH}
> Summarize this into a CSV.

Style Instructions Style instructions are a type of output formatting used to modify the output stylistically rather than structurally (Section 2.2.1.3). For example:

> Write a clear and curt paragraph about llamas.

Role A Role, also known as a persona (Schmidt et al., 2023; Wang et al., 2023l), is a frequently discussed component that can improve writing and style text (Section 2.2.1.3). For example:

> Pretend you are a shepherd and write a limerick about llamas.

Additional Information It is often necessary to include additional information in the prompt. For example, if the directive is to write an email, you might include information such as your name and position so the GenAI can properly sign the email. Additional Information is sometimes called 'context', though we discourage the use of this term as it is overloaded with other meanings in the prompting space[2].

1.2.2 Prompting Terms

Terminology within the prompting literature is rapidly developing. As it stands, there are many poorly understood definitions (e.g. prompt, prompt engineering) and conflicting ones (e.g. role prompt vs persona prompt). The lack of a consistent vocabulary hampers the community's ability to clearly describe the various prompting techniques in use. We provide a robust vocabulary of terms used in the prompting community (Figure 1.3).[3] Less frequent terms are left to Appendix A.2. In order to accurately define frequently-used terms like prompt and prompt engineering, we integrate many definitions (Appendix A.1) to derive representative definitions.

Prompting Prompting is the process of providing a prompt to a GenAI, which then generates a response. For example, the action of sending a

[2]e.g. the context is the tokens processed by the LLM in a forward pass

[3]By robust, we mean that it covers most existing commonly used terms in the field.

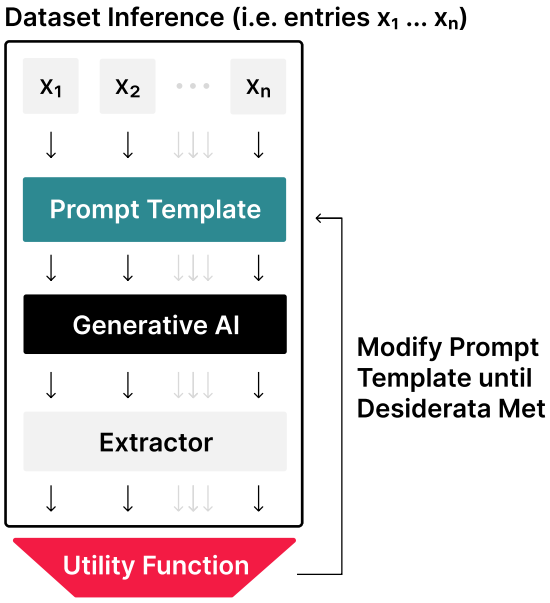

Figure 1.4: The Prompt Engineering Process consists of three repeated steps 1) performing inference on a dataset 2) evaluating performance and 3) modifying the prompt template. Note that the extractor is used to extract a final response from the LLM output (e.g. "This phrase is positive" → "positive"). See more information on extractors in Section 2.5.

chunk of text or uploading an image constitutes prompting.

Prompt Chain A prompt chain (activity: prompt chaining) consists of two or more prompt templates used in succession. The output of the prompt generated by the first prompt template is used to parameterize the second template, continuing until all templates are exhausted (Wu et al., 2022).

Prompting Technique A prompting technique is a blueprint that describes how to structure a prompt, prompts, or dynamic sequencing of multiple prompts. A prompting technique may incorporate conditional or branching logic, parallelism, or other architectural considerations spanning multiple prompts.

Prompt Engineering Prompt engineering is the iterative process of developing a prompt by modifying or changing the prompting technique that you are using (Figure 1.4).

Prompt Engineering Technique A prompt engineering technique is a strategy for iterating on a prompt to improve it. In literature, this will often be automated techniques (Deng et al., 2022), but in consumer settings, users often perform prompt engineering manually, without any assistive tooling.

Exemplar Exemplars are examples of a task being completed that are shown to a model in a prompt (Brown et al., 2020).

1.3 A Short History of Prompts

The idea of using natural language prefixes, or prompts, to elicit language model behaviors and responses originated before the GPT-3 and ChatGPT era. GPT-2 (Radford et al., 2019a) makes use of prompts and they appear to be first used in the context of Generative AI by Fan et al. (2018). However, the concept of prompts was preceded by related concepts such as control codes (Pfaff, 1979; Poplack, 1980; Keskar et al., 2019) and writing prompts in literature.

The term Prompt Engineering appears to have come into existence more recently from Radford et al. (2021) then slightly later from Reynolds and McDonell (2021).

However, various papers perform prompt engineering without naming the term (Wallace et al., 2019; Shin et al., 2020a), including Schick and Schütze (2020a,b); Gao et al. (2021) for non-autoregressive language models.

Some of the first works on prompting define a prompt slightly differently to how it is currently used. For example, consider the following prompt from Brown et al. (2020):

> Translate English to French:
> llama

Brown et al. (2020) consider the word "llama" to be the prompt, while "Translate English to French:" is the "task description". More recent papers, including this one, refer to the entire string passed to the LLM as the prompt.

2 A Meta-Analysis of Prompting

2.1 Systematic Review Process

In order to robustly collect a dataset of sources for this paper, we ran a systematic literature review grounded in the PRISMA process (Page et al., 2021) (Figure 2.1). We host this dataset on HuggingFace [4] and present a datasheet (Gebru et al., 2021) for the dataset in Appendix A.3. Our main data sources were arXiv, Semantic Scholar, and ACL. We query these databases with a list of 44 keywords narrowly related to prompting and prompt engineering (Appendix A.4).

2.1.1 The Pipeline

In this section, we introduce our data scraping pipeline, which includes both human and LLM-assisted review.[5] As an initial sample to establish filtering critera, we retrieve papers from arXiv based on a simple set of keywords and boolean rules (A.4). Then, human annotators label a sample of 1,661 articles from the arXiv set for the following criteria:

1. Include if the paper proposes a novel prompting technique.

2. Include if the paper strictly covers hard prefix prompts.

3. Exclude if the paper focuses on training by backpropagating gradients.

4. Include if the paper uses a masked frame and/or window for non-text modalities.

A set of 300 articles are reviewed independently by two annotators, with 92% agreement (Krippendorff's α = Cohen's κ = 81%). Next, we develop a prompt using gpt-4-1106-preview to classify the remaining articles (Appendix A.5). We validate the prompt against 100 ground-truth annotations, achieving 89% precision and 75% recall (for an $F1$ of 81%). The combined human and LLM annotations generate a final set of 1,565 papers.

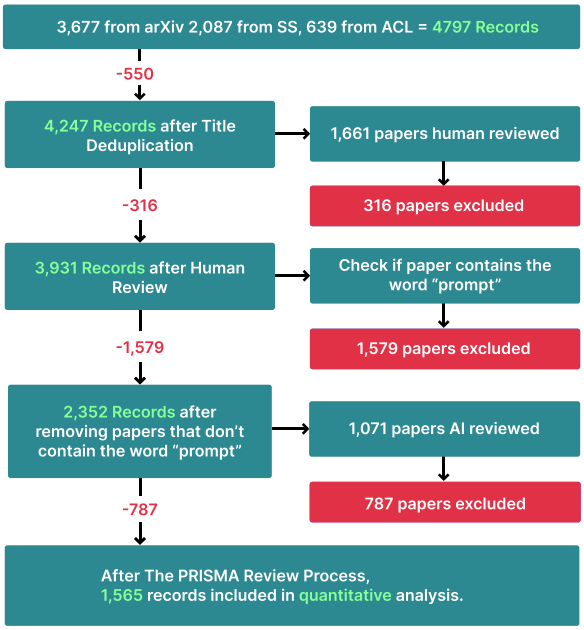

Figure 2.1: The PRISMA systematic literature review process. We accumulate 4,247 unique records from which we extract 1,565 relevant records.

2.2 Text-Based Techniques

We now present a comprehensive taxonomical ontology of 58 text-based prompting techniques, broken into 6 major categories (Figure 2.2). Although some of the techniques might fit into multiple categories, we place them in a single category of most relevance.

2.2.1 In-Context Learning (ICL)

ICL refers to the ability of GenAIs to learn skills and tasks by providing them with exemplars and or relevant instructions within the prompt, without the need for weight updates/retraining (Brown et al., 2020; Radford et al., 2019b). These skills can be learned from exemplars (Figure 2.4) and/or instructions (Figure 2.5). Note that the word "learn" is misleading. ICL can simply be task specification–the skills are not necessarily new, and can have already been included in the training data (Figure 2.6). See Appendix A.9 for a discussion of the use of this term. Significant work is currently being done on optimizing (Bansal et al., 2023) and understanding (Si et al., 2023a; Štefánik and Kadlčík, 2023) ICL.

[4] https://huggingface.co/datasets/PromptSystematicReview/Prompt_Systematic_Review_Dataset
[5] Using gpt-4-1106-preview

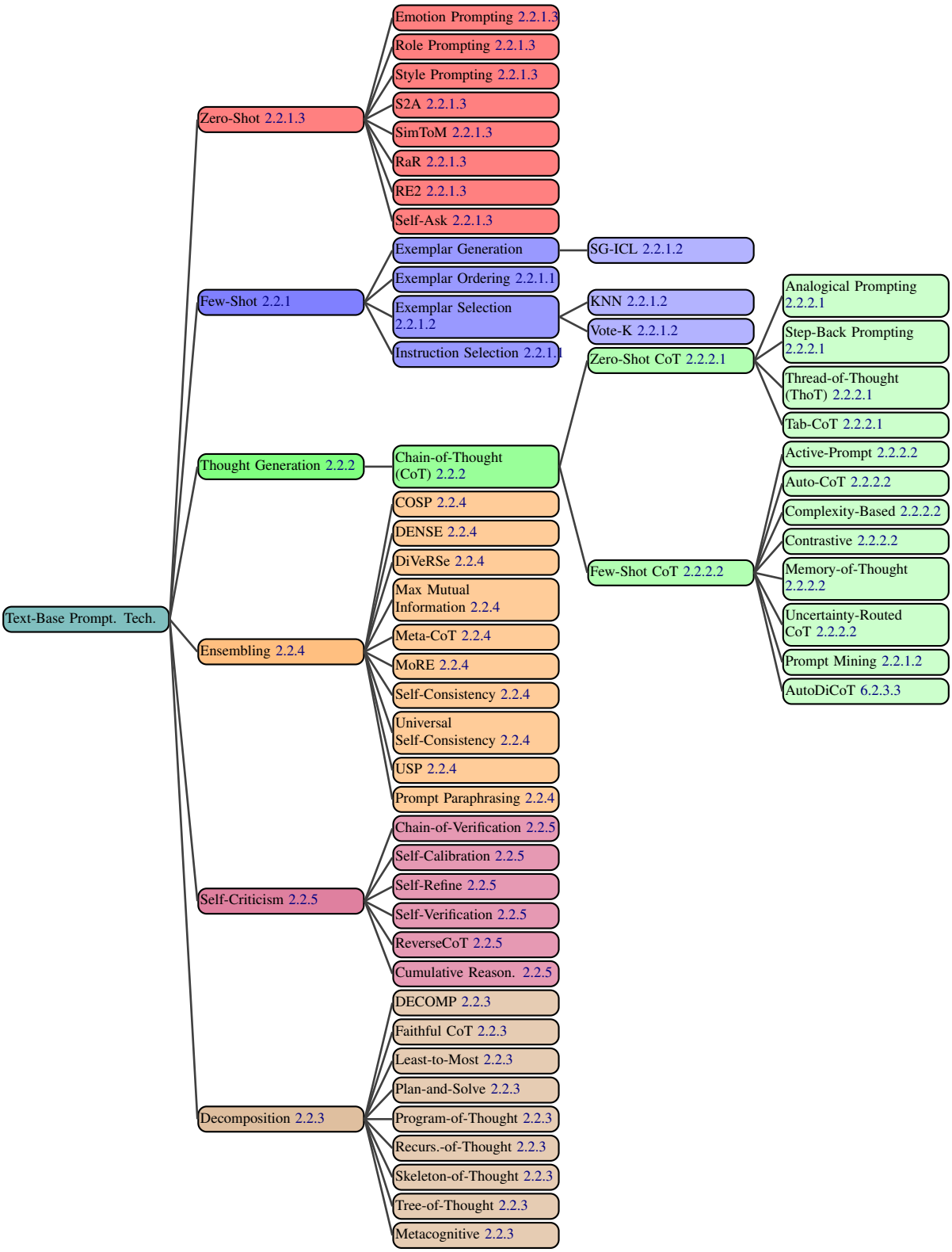

Figure 2.2: All text-based prompting techniques from our dataset.

Figure 2.3: We highlight six main design decisions when crafting few-shot prompts. *Please note that recommendations here *do not* generalize to all tasks; in some cases, each of them could hurt performance.

Few-Shot Prompting (Brown et al., 2020) is the paradigm seen in Figure 2.4, where the GenAI learns to complete a task with only a few examples (exemplars). Few-shot prompting is a special case of Few-Shot Learning (FSL) (Fei-Fei et al., 2006; Wang et al., 2019), but does not require updating of model parameters

2.2.1.1 Few-Shot Prompting Design Decisions

Selecting exemplars for a prompt is a difficult task–performance depends significantly on various factors of the exemplars (Dong et al., 2023), and only a limited number of exemplars fit in the typical LLM's context window. We highlight six separate design decisions, including the selection and order of exemplars that critically influence the output quality (Zhao et al., 2021a; Lu et al., 2021; Ye and Durrett, 2023) (Figure 2.3).

```
2+2: four
4+5: nine
8+0:
```

Figure 2.4: ICL exemplar prompt

```
Extract all words that have 3 of the same
letter and at least 3 other letters from the
following text: {TEXT}
```

Figure 2.5: ICL instruction prompt

Exemplar Quantity Increasing the quantity of exemplars in the prompt generally improves model performance, particularly in larger models (Brown et al., 2020). However, in some cases, the benefits may diminish beyond 20 exemplars (Liu et al., 2021). In the case of long context LLMs, additional exemplars continue to increase performance, though efficiency varies depending on task and model (Agarwal et al., 2024; Bertsch et al., 2024; Jiang et al., 2024).

Exemplar Ordering The order of exemplars affects model behavior (Lu et al., 2021; Kumar and Talukdar, 2021; Liu et al., 2021; Rubin et al., 2022). On some tasks, exemplar order can cause accuracy to vary from sub-50% to 90%+ (Lu et al., 2021).

Exemplar Label Distribution As in traditional supervised machine learning, the distribution of exemplar labels in the prompt affects behavior. For example, if 10 exemplars from one class and 2 exemplars of another class are included, this may cause the model to be biased toward the first class.

Exemplar Label Quality Despite the general benefit of multiple exemplars, the necessity of strictly *valid* demonstrations is unclear. Some work (Min et al., 2022) suggests that the accuracy of labels is irrelevant—providing models with exemplars with incorrect labels may not negatively diminish performance. However, under certain settings, there is a significant impact on performance (Yoo et al., 2022). Larger models are often better at handling incorrect or unrelated labels (Wei et al., 2023c).

It is important to discuss this factor, since if you are automatically constructing prompts from large datasets that may contain inaccuracies, it may be

Translate the word "cheese" to French.

Figure 2.6: ICL from training data prompt. In this version of ICL, the model is not learning a new skill, but rather using knowledge likely in its training set.

necessary to study how label quality affects your results.

Exemplar Format The formatting of exemplars also affects performance. One of the most common formats is "Q: {input}, A: {label}", but the optimal format may vary across tasks; it may be worth trying multiple formats to see which performs best. There is some evidence to suggest that formats that occur commonly in the training data will lead to better performance (Jiang et al., 2020).

Exemplar Similarity Selecting exemplars that are similar to the test sample is generally beneficial for performance (Liu et al., 2021; Min et al., 2022). However, in some cases, selecting more diverse exemplars can improve performance (Su et al., 2022; Min et al., 2022).

Instruction Selection While instructions are required to guide LLMs in zero-shot prompts (Wei et al., 2022a), the benefits of adding instructions before exemplars in few-shot prompts is less clear. Ajith et al. (2024) show that generic, task-agnostic instructions (i.e., no instruction or "Complete the following task:") improve classification and question answering accuracy over task-specific ones (e.g., What is the answer to this question?) concluding instruction-following abilities can be achieved via exemplars alone. While they may not improve correctness, instructions in few-shot prompts can still guide auxiliary output attributes like writing style (Roy et al., 2023).

2.2.1.2 Few-Shot Prompting Techniques

Considering all of these factors, Few-Shot Prompting can be very difficult to implement effectively. We now examine techniques for Few-Shot Prompting in the supervised setting. Ensembling approaches can also benefit Few-Shot Prompting, but we discuss them separately (Section 2.2.4).

Assume we have a training dataset, D^{train}, which contains multiple inputs $D^{train}_{x^i}$ and outputs $D^{train}_{y^i}$, which can be used to few-shot prompt a GenAI (rather than performing gradient-based updates). Assume that this prompt can be dynamically generated with respect to $D^{test}_{x^i}$ at test time. Here is the prompt template we will use for this section, following the 'input: output' format (Figure 2.4):

{Exemplars}
$D^{test}_{x^i}$:

Figure 2.7: Few-Shot Prompting Template

K-Nearest Neighbor (KNN) (Liu et al., 2021) is part of a family of algorithms that selects exemplars similar to $D^{test}_{x^i}$ to boost performance. Although effective, employing KNN during prompt generation may be time and resource intensive.

Vote-K (Su et al., 2022) is another method to select similar exemplars to the test sample. In one stage, a model proposes useful unlabeled candidate exemplars for an annotator to label. In the second stage, the labeled pool is used for Few-Shot Prompting. Vote-K also ensures that newly added exemplars are sufficiently different than existing ones to increase diversity and representativeness.

Self-Generated In-Context Learning (SG-ICL) (Kim et al., 2022) leverages a GenAI to automatically generate exemplars. While better than zero-shot scenarios when training data is unavailable, the generated samples are not as effective as actual data.

Prompt Mining (Jiang et al., 2020) is the process of discovering optimal "middle words" in prompts through large corpus analysis. These middle words are effectively prompt templates. For example, instead of using the common "Q: A:" format for few-shot prompts, there may exist something similar that occurs more frequently in the corpus. Formats which occur more often in the corpus will likely lead to improved prompt performance.

More Complicated Techniques such as LENS (Li and Qiu, 2023a), UDR (Li et al., 2023f), and Active Example Selection (Zhang et al., 2022a) leverage iterative filtering, embedding and retrieval, and reinforcement learning, respectively.

2.2.1.3 Zero-Shot Prompting Techniques

In contrast to Few-Shot Prompting, Zero-Shot Prompting uses zero exemplars. There are a number of well-known standalone zero-shot techniques

as well as zero-shot techniques combined with another concept (e.g. Chain of Thought), which we discuss later (Section 2.2.2).

Role Prompting (Wang et al., 2023j; Zheng et al., 2023d) , also known as persona prompting (Schmidt et al., 2023; Wang et al., 2023l), assigns a specific role to the GenAI in the prompt. For example, the user might prompt it to act like "Madonna" or a "travel writer". This can create more desirable outputs for open-ended tasks (Reynolds and McDonell, 2021) and in some cases may improve accuracy on benchmarks (Zheng et al., 2023d).

Style Prompting (Lu et al., 2023a) involves specifying the desired style, tone, or genre in the prompt to shape the output of a GenAI. A similar effect can be achieved using role prompting.

Emotion Prompting (Li et al., 2023a) incorporates phrases of psychological relevance to humans (e.g., "This is important to my career") into the prompt, which may lead to improved LLM performance on benchmarks and open-ended text generation.

System 2 Attention (S2A) (Weston and Sukhbaatar, 2023) first asks an LLM to rewrite the prompt and remove any information unrelated to the question therein. Then, it passes this new prompt into an LLM to retrieve a final response.

SimToM (Wilf et al., 2023) deals with complicated questions which involve multiple people or objects. Given the question, it attempts to establish the set of facts one person knows, then answer the question based only on those facts. This is a two prompt process and can help eliminate the effect of irrelevant information in the prompt.

Rephrase and Respond (RaR) (Deng et al., 2023) instructs the LLM to rephrase and expand the question before generating the final answer. For example, it might add the following phrase to the question: "Rephrase and expand the question, and respond". This could all be done in a single pass or the new question could be passed to the LLM separately. RaR has demonstrated improvements on multiple benchmarks.

Re-reading (RE2) (Xu et al., 2023) adds the phrase "Read the question again:" to the prompt in addition to repeating the question. Although this is such a simple technique, it has shown improvement in reasoning benchmarks, especially with complex questions.

> Q: Jack has two baskets, each containing three balls. How many balls does Jack have in total?
> A: One basket contains 3 balls, so two baskets contain 3 * 2 = 6 balls.
> Q: {QUESTION}
> A:

Figure 2.8: A One-Shot Chain-of-Thought Prompt.

Self-Ask (Press et al., 2022) prompts LLMs to first decide if they need to ask follow up questions for a given prompt. If so, the LLM generates these questions, then answers them and finally answers the original question.

2.2.2 Thought Generation

Thought generation encompasses a range of techniques that prompt the LLM to articulate its reasoning while solving a problem (Zhang et al., 2023c).

Chain-of-Thought (CoT) Prompting (Wei et al., 2022b) leverages few-shot prompting to encourage the LLM to express its thought process before delivering its final answer.[6] This technique is occasionally referred to as Chain-of-Thoughts (Tutunov et al., 2023; Besta et al., 2024; Chen et al., 2023d). It has been demonstrated to significantly enhance the LLM's performance in mathematics and reasoning tasks. In Wei et al. (2022b), the prompt includes an exemplar featuring a question, a reasoning path, and the correct answer (Figure 2.8).

2.2.2.1 Zero-Shot-CoT

The most straightforward version of CoT contains zero exemplars. It involves appending a thought inducing phrase like "Let's think step by step." (Kojima et al., 2022) to the prompt. Other suggested thought-generating phrases include "First, let's think about this logically" (Kojima et al., 2022). Zhou et al. (2022b) uses LLMs to generate "Let's work this out in a step by step way to be sure we have the right answer". Yang et al. (2023a) searches for an optimal thought inducer. Zero-Shot-

[6] We note that such techniques are often described using words like "think" that anthropomorphize models. We attempt not to use this language, but do use original authors' language where appropriate.

CoT approaches are attractive as they don't require exemplars and are generally task agnostic.

Step-Back Prompting (Zheng et al., 2023c) is a modification of CoT where the LLM is first asked a generic, high-level question about relevant concepts or facts before delving into reasoning. This approach has improved performance significantly on multiple reasoning benchmarks for both PaLM-2L and GPT-4.

Analogical Prompting (Yasunaga et al., 2023) is similar to SG-ICL, and automatically generates exemplars that include CoTs. It has demonstrated improvements in mathematical reasoning and code generation tasks.

Thread-of-Thought (ThoT) Prompting (Zhou et al., 2023) consists of an improved thought inducer for CoT reasoning. Instead of "Let's think step by step," it uses "Walk me through this context in manageable parts step by step, summarizing and analyzing as we go." This thought inducer works well in question-answering and retrieval settings, especially when dealing with large, complex contexts.

Tabular Chain-of-Thought (Tab-CoT) (Jin and Lu, 2023) consists of a Zero-Shot CoT prompt that makes the LLM output reasoning as a markdown table. This tabular design enables the LLM to improve the structure and thus the reasoning of its output.

2.2.2.2 Few-Shot CoT

This set of techniques presents the LLM with multiple exemplars, which include chains-of-thought. This can significantly enhance performance. This technique is occasionally referred to as Manual-CoT (Zhang et al., 2022b) or Golden CoT (Del and Fishel, 2023).

Contrastive CoT Prompting (Chia et al., 2023) adds both exemplars with incorrect and correct explanations to the CoT prompt in order to show the LLM how *not* to reason. This method has shown significant improvement in areas like Arithmetic Reasoning and Factual QA.

Uncertainty-Routed CoT Prompting (Google, 2023) samples multiple CoT reasoning paths, then selects the majority if it is above a certain threshold (calculated based on validation data). If not, it samples greedily and selects that response. This method demonstrates improvement on the MMLU benchmark for both GPT-4 and Gemini Ultra models.

Complexity-based Prompting (Fu et al., 2023b) involves two major modifications to CoT. First, it selects complex examples for annotation and inclusion in the prompt, based on factors like question length or reasoning steps required. Second, during inference, it samples multiple reasoning chains (answers) and uses a majority vote among chains exceeding a certain length threshold, under the premise that longer reasoning indicates higher answer quality. This technique has shown improvements on three mathematical reasoning datasets.

Active Prompting (Diao et al., 2023) starts with some training questions/exemplars, asks the LLM to solve them, then calculates uncertainty (disagreement in this case) and asks human annotators to rewrite the exemplars with highest uncertainty.

Memory-of-Thought Prompting (Li and Qiu, 2023b) leverage unlabeled training exemplars to build Few-Shot CoT prompts at test time. Before test time, it performs inference on the unlabeled training exemplars with CoT. At test time, it retrieves similar instances to the test sample. This technique has shown substantial improvements in benchmarks like Arithmetic, commonsense, and factual reasoning.

Automatic Chain-of-Thought (Auto-CoT) Prompting (Zhang et al., 2022b) uses Wei et al. (2022b)'s Zero-Shot prompt to automatically generate chains of thought. These are then used to build a Few-Shot CoT prompt for a test sample.

2.2.3 Decomposition

Significant research has focused on decomposing complex problems into simpler sub-questions. This is an effective problem-solving strategy for humans as well as GenAI (Patel et al., 2022). Some decomposition techniques are similar to thought-inducing techniques, such as CoT, which often naturally breaks down problems into simpler components. However, explicitly breaking down problems can further improve LLMs' problem solving ability.

Least-to-Most Prompting (Zhou et al., 2022a) starts by prompting a LLM to break a given problem into sub-problems without solving them. Then, it solves them sequentially, appending model responses to the prompt each time, until it arrives

at a final result. This method has shown significant improvements in tasks involving symbolic manipulation, compositional generalization, and mathematical reasoning.

Decomposed Prompting (DECOMP) (Khot et al., 2022) Few-Shot prompts a LLM to show it how to use certain functions. These might include things like string splitting or internet searching; these are often implemented as separate LLM calls. Given this, the LLM breaks down its original problem into sub-problems which it sends to different functions. It has shown improved performance over Least-to-Most prompting on some tasks.

Plan-and-Solve Prompting (Wang et al., 2023f) consists of an improved Zero-Shot CoT prompt, "Let's first understand the problem and devise a plan to solve it. Then, let's carry out the plan and solve the problem step by step". This method generates more robust reasoning processes than standard Zero-Shot-CoT on multiple reasoning datasets.

Tree-of-Thought (ToT) (Yao et al., 2023b), also known as Tree of Thoughts, (Long, 2023), creates a tree-like search problem by starting with an initial problem then generating multiple possible steps in the form of thoughts (as from a CoT). It evaluates the progress each step makes towards solving the problem (through prompting) and decides which steps to continue with, then keeps creating more thoughts. ToT is particularly effective for tasks that require search and planning.

Recursion-of-Thought (Lee and Kim, 2023) is similar to regular CoT. However, every time it encounters a complicated problem in the middle of its reasoning chain, it sends this problem into another prompt/LLM call. After this is completed, the answer is inserted into the original prompt. In this way, it can recursively solve complex problems, including ones which might otherwise run over that maximum context length. This method has shown improvements on arithmetic and algorithmic tasks. Though implemented using fine-tuning to output a special token that sends sub-problem into another prompt, it could also be done only through prompting.

Program-of-Thoughts (Chen et al., 2023d) uses LLMs like Codex to generate programming code as reasoning steps. A code interpreter executes these steps to obtain the final answer. It excels in mathematical and programming-related tasks but is less effective for semantic reasoning tasks.

Faithful Chain-of-Thought (Lyu et al., 2023) generates a CoT that has both natural language and symbolic language (e.g. Python) reasoning, just like Program-of-Thoughts. However, it also makes use of different types of symbolic languages in a task-dependent fashion.

Skeleton-of-Thought (Ning et al., 2023) focuses on accelerating answer speed through parallelization. Given a problem, it prompts an LLM to create a skeleton of the answer, in a sense, sub-problems to be solved. Then, in parallel, it sends these questions to a LLM and concatenates all the outputs to get a final response.

Metacognitive Prompting (Wang and Zhao, 2024) attempts to make the LLM mirror human metacognitive processes with a five part prompt chain, with steps including clarifying the question, preliminary judgement, evaluation of response, decision confirmation, and confidence assessment.

2.2.4 Ensembling

In GenAI, ensembling is the process of using multiple prompts to solve the same problem, then aggregating these responses into a final output. In many cases, a majority vote—selecting the most frequent response—is used to generate the final output. Ensembling techniques reduce the variance of LLM outputs and often improving accuracy, but come with the cost of increasing the number of model calls needed to reach a final answer.

Demonstration Ensembling (DENSE) (Khalifa et al., 2023) creates multiple few-shot prompts, each containing a distinct subset of exemplars from the training set. Next, it aggregates over their outputs to generate a final response.

Mixture of Reasoning Experts (MoRE) (Si et al., 2023d) creates a set of diverse reasoning experts by using different specialized prompts for different reasoning types (such as retrieval augmentation prompts for factual reasoning, Chain-of-Thought reasoning for multi-hop and math reasoning, and generated knowledge prompting for commonsense reasoning). The best answer from all experts is selected based on an agreement score.

Max Mutual Information Method (Sorensen et al., 2022) creates multiple prompt templates with

varied styles and exemplars, then selects the optimal template as the one that maximizes mutual information between the prompt and the LLM's outputs.

Self-Consistency (Wang et al., 2022) is based on the intuition that multiple different reasoning paths can lead to the same answer. This method first prompts the LLM multiple times to perform CoT, crucially with a non-zero temperature to elicit diverse reasoning paths. Next, it uses a majority vote over all generated responses to select a final response. Self-Consistency has shown improvements on arithmetic, commonsense, and symbolic reasoning tasks.

Universal Self-Consistency (Chen et al., 2023e) is similar to Self-Consistency except that rather than selecting the majority response by programmatically counting how often it occurs, it inserts all outputs into a prompt template that selects the majority answer. This is helpful for free-form text generation and cases where the same answer may be output slightly differently by different prompts.

Meta-Reasoning over Multiple CoTs (Yoran et al., 2023) is similar to universal Self-Consistency; it first generates multiple reasoning chains (but not necessarily final answers) for a given problem. Next, it inserts all of these chains in a single prompt template then generates a final answer from them.

DiVeRSe (Li et al., 2023i) creates multiple prompts for a given problem then performs Self-Consistency for each, generating multiple reasoning paths. They score reasoning paths based on each step in them then select a final response.

Consistency-based Self-adaptive Prompting (COSP) (Wan et al., 2023a) constructs Few-Shot CoT prompts by running Zero-Shot CoT with Self-Consistency on a set of examples then selecting a high agreement subset of the outputs to be included in the final prompt as exemplars. It again performs Self-Consistency with this final prompt.

Universal Self-Adaptive Prompting (USP) (Wan et al., 2023b) builds upon the success of COSP, aiming to make it generalizable to all tasks. USP makes use of unlabeled data to generate exemplars and a more complicated scoring function to select them. Additionally, USP does not use Self-Consistency.

Prompt Paraphrasing (Jiang et al., 2020) transforms an original prompt by changing some of the wording, while still maintaining the overall meaning. It is effectively a data augmentation technique that can be used to generate prompts for an ensemble.

2.2.5 Self-Criticism

When creating GenAI systems, it can be useful to have LLMs criticize their own outputs (Huang et al., 2022). This could simply be a judgement (e.g., is this output correct) or the LLM could be prompted to provide feedback, which is then used to improve the answer. Many approaches to generating and integrating self-criticism have been developed.

Self-Calibration (Kadavath et al., 2022) first prompts an LLM to answer a question. Then, it builds a new prompt that includes the question, the LLM's answer, and an additional instruction asking whether the answer is correct. This can be useful for gauging confidence levels when applying LLMs when deciding when to accept or revise the original answer.

Self-Refine (Madaan et al., 2023) is an iterative framework where, given an initial answer from the LLM, it prompts the same LLM to provide feedback on the answer, and then prompts the LLM to improve the answer based on the feedback. This iterative process continues until a stopping condition is met (e.g., max number of steps reached). Self-Refine has demonstrated improvement across a range of reasoning, coding, and generation tasks.

Reversing Chain-of-Thought (RCoT) (Xue et al., 2023) first prompts LLMs to reconstruct the problem based on generated answer. Then, it generates fine-grained comparisons between the original problem and the reconstructed problem as a way to check for any inconsistencies. These inconsistencies are then converted to feedback for the LLM to revise the generated answer.

Self-Verification (Weng et al., 2022) generates multiple candidate solutions with Chain-of-Thought (CoT). It then scores each solution by masking certain parts of the original question and asking an LLM to predict them based on the rest of the question and the generated solution. This method has shown improvement on eight reasoning datasets.

Chain-of-Verification (COVE) (Dhuliawala et al., 2023) first uses an LLM to generate an answer to a given question. Then, it creates a list of related questions that would help verify the correctness of the answer. Each question is answered by the LLM, then all the information is given to the LLM to produce the final revised answer. This method has shown improvements in various question-answering and text-generation tasks.

Cumulative Reasoning (Zhang et al., 2023b) first generates several potential steps in answering the question. It then has a LLM evaluate them, deciding to either accept or reject these steps. Finally, it checks whether it has arrived at the final answer. If so, it terminates the process, but otherwise it repeats it. This method has demonstrated improvements in logical inference tasks and mathematical problem.

2.3 Prompting Technique Usage

As we have just seen, there exist many text-based prompting techniques. However, only a small subset of them are commonly used in research and in industry. We measure technique usage by proxy of measuring the number of citations by other papers in our dataset. We do so with the presumption that papers about prompting are more likely to actually use or evaluate the cited technique. We graph the top 25 papers cited in this way from our dataset and find that most of them propose new prompting techniques (Figure 2.11). The prevalence of citations for Few-Shot and Chain-of-Thought prompting is unsurprising and helps to establish a baseline for understanding the prevalence of other techniques.

2.3.1 Benchmarks

In prompting research, when researchers propose a new technique, they usually benchmark it across multiple models and datasets. This is important to prove the utility of the technique and examine how it transfers across models.

In order to make it easier for researchers proposing new techniques to know how to benchmark them, we quantitatively examine which models (Figure 2.9) and what benchmark datasets (Figure 2.10) are being used. Again, we measure usage by how many times papers in our dataset cite the benchmark datasets and models.

To find which datasets and models are being used, we prompted GPT-4-1106-preview to extract any mentioned dataset or model from the body of papers in our dataset. After, we manually filtered out results that were not models or datasets. The citation counts were acquired by searching items from the finalized list on Semantic Scholar.

2.4 Prompt Engineering

In addition to surveying prompting techniques, we also review prompt engineering techniques, which are used to automatically optimize prompts. We discuss some techniques that use gradient updates, since the set of prompt engineering techniques is much smaller than that of prompting techniques.

Meta Prompting is the process of prompting a LLM to generate or improve a prompt or prompt template (Reynolds and McDonell, 2021; Zhou et al., 2022b; Ye et al., 2023). This is often done without any scoring mechanism, using just a simple template (Figure 2.12). However, other works present more complex uses of meta-prompting, with multiple iterations and scoring mechanisms Yang et al. (2023a); Fernando et al. (2023).

> Improve the following prompt: {PROMPT}

Figure 2.12: A simple Meta Prompting template.

AutoPrompt (Shin et al., 2020b) uses a frozen LLM as well as a prompt template that includes some "trigger tokens", whose values are updated via backpropogation at training time. This is a version of soft-prompting.

Automatic Prompt Engineer (APE) (Zhou et al., 2022b) uses a set of exemplars to generate a Zero-Shot instruction prompt. It generates multiple possible prompts, scores them, then creates variations of the best ones (e.g. by using prompt paraphrasing). It iterates on this process until some desiderata are reached.

Gradientfree Instructional Prompt Search (GrIPS) (Prasad et al., 2023) is similar to APE, but uses a more complex set of operations including deletion, addition, swapping, and paraphrasing in order to create variations of a starting prompt.

Prompt Optimization with Textual Gradients (ProTeGi) (Pryzant et al., 2023) is a unique approach to prompt engineering that improves a prompt template through a multi-step process. First, it passes

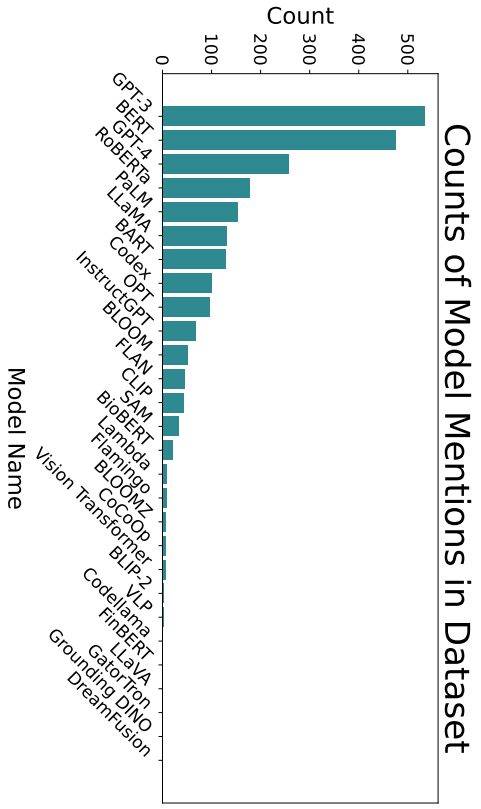

Figure 2.9: Citation Counts of GenAI Models

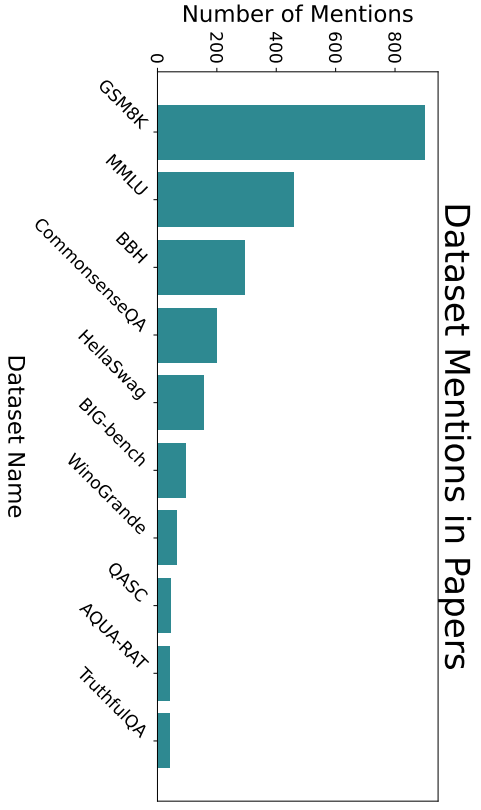

Figure 2.10: Citation Counts of Datasets

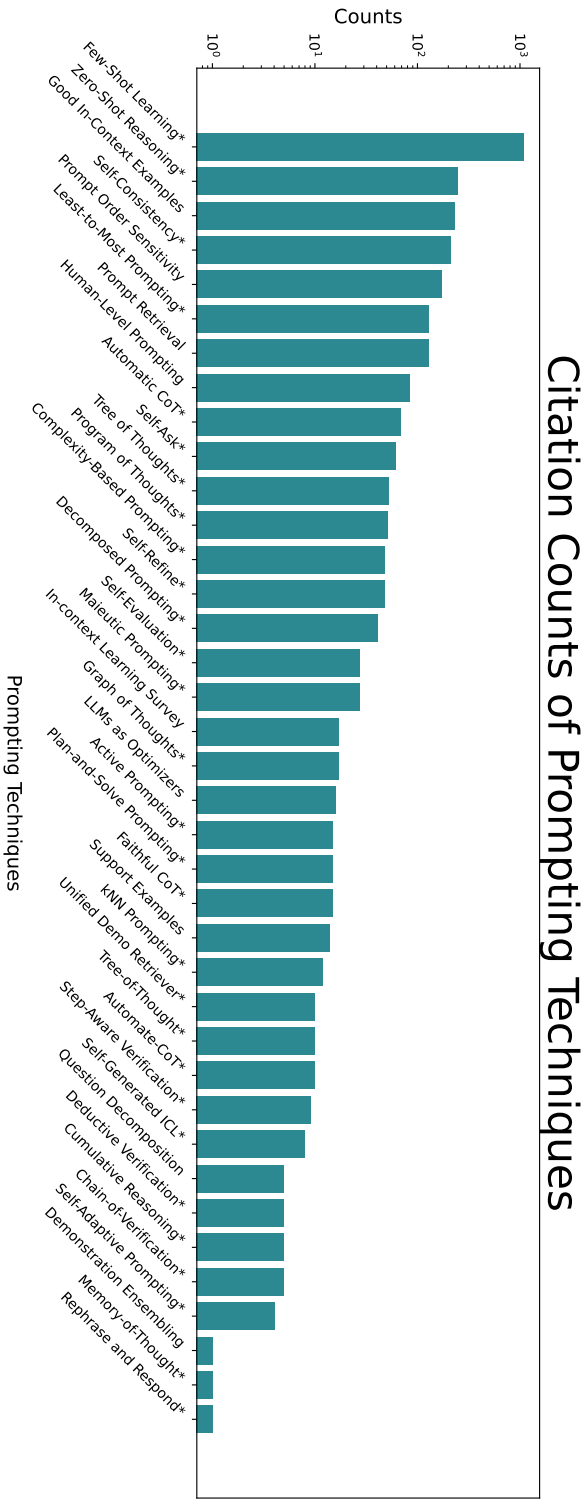

Figure 2.11: Citation Counts of Prompting Techniques. The top 25 papers in our dataset, measured by how often they are cited by other papers in our dataset. Most papers here are prompting techniques*, and the remaining papers contains prompting advice.

a batch of inputs through the template, then passes the output, ground truth, and prompt into another prompt that criticizes the original prompt. It generates new prompts from these criticisms then uses a bandit algorithm (Gabillon et al., 2011) to select one. ProTeGi demonstrates improvements over methods like APE and GRIPS.

RLPrompt (Deng et al., 2022) uses a frozen LLM with an unfrozen module added. It uses this LLM to generate prompt templates, scores the templates on a dataset, and updates the unfrozen module using Soft Q-Learning (Guo et al., 2022). Interestingly, the method often selects grammatically nonsensical text as the optimal prompt template.

Dialogue-comprised Policy-gradient-based Discrete Prompt Optimization (DP2O) (Li et al., 2023b) is perhaps the most complicated prompt engineering technique, involving reinforcement learning, a custom prompt scoring function, and conversations with an LLM to construct the prompt.

2.5 Answer Engineering

Answer engineering is the iterative process of developing or selecting among algorithms that extract precise answers from LLM outputs. To understand the need for answer engineering, consider a binary classification task where the labels are "Hate Speech" and "Not Hate Speech". The prompt template might look like this:

> Is this "Hate Speech" or "Not Hate Speech":
> {TEXT}

When a hate speech sample is put through the template, it might have outputs such as "It's hate speech", "Hate Speech.", or even "Hate speech, because it uses negative language against a racial group". This variance in response formats is difficult to parse consistently; improved prompting can help, but only to a certain extent.

There are three design decisions in answer engineering, the choice of answer space, answer shape, and answer extractor (Figure 2.13). Liu et al. (2023b) define the first two as necessary components of answer engineering and we append the third. We consider answer engineering to be distinct from prompt engineering, but extremely closely related; the processes are often conducted in tandem.

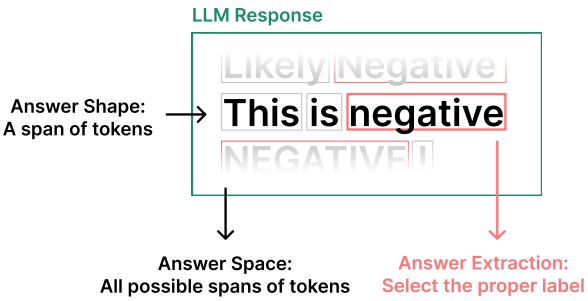

Figure 2.13: An annotated output of a LLM output for a labeling task, which shows the three design decisions of answer engineering: the choice of answer shape, space, and extractor. Since this is an output from a classification task, the answer shape could be restricted to a single token and the answer space to one of two tokens ("positive" or "negative"), though they are unrestricted in this image.

2.5.1 Answer Shape

The shape of an answer is its physical format. For example, it could be a token, span of tokens, or even an image or video.[7] It is sometimes useful to restrict the output shape of a LLM to a single token for tasks like binary classification.

2.5.2 Answer Space

The space of an answer is the domain of values that its structure may contain. This may simply be the space of all tokens, or in a binary labeling task, could just be two possible tokens.

2.5.3 Answer Extractor

In cases where it is impossible to entirely control the answer space (e.g. consumer-facing LLMs), or the expected answer may be located somewhere within the model output, a rule can be defined to extract the final answer. This rule is often a simple function (e.g. a regular expression), but can also use a separate LLM to extract the answer.

Verbalizer Often used in labeling tasks, a verbalizer maps a token, span, or other type of output to a label and vice-versa (injective) (Schick and Schütze, 2021). For example, if we wish for a model to predict whether a Tweet is positive or negative, we could prompt it to output either "+" or "-" and a verbalizer would map these token sequences to the appropriate labels. The selection of a verbalizer constitutes a component of answer engineering.

[7] We use a different definition than Liu et al. (2023b) with respect to granularity (e.g. token vs span), since the output could be of a different modality.

Regex As mentioned previously, Regexes are often used to extract answers. They are usually used to search for the first instance of a label. However, depending on the output format and whether CoTs are generated, it may be better to search for the last instance.

Separate LLM Sometimes outputs are so complicated that regexes won't work consistently. In this case, it can be useful to have a separate LLM evaluate the output and extract an answer. This separate LLM will often use an answer trigger (Kojima et al., 2022), e.g. "The answer (Yes or No) is", to extract the answer.

3 Beyond English Text Prompting

Prompting GenAIs with English text currently stands as the dominant method for interaction. Prompting in other languages or through different modalities often requires special techniques to achieve comparable performance. In this context, we discuss the domains of multilingual and multimodal prompting.

3.1 Multilingual

State-of-the-art GenAIs have often been predominately trained with English dataset, leading to a notable disparity in the output quality in languages other than English, particularly low-resource languages (Bang et al., 2023; Jiao et al., 2023; Hendy et al., 2023; Shi et al., 2022). As a result, various multilingual prompting techniques have emerged in an attempt to improve model performance in non-English settings (Figure 3.1).

Translate First Prompting (Shi et al., 2022) is perhaps the simplest strategy and first translates non-English input examples into English. By translating the inputs into English, the model can utilize its strengths in English to better understand the content. Translation tools vary; Shi et al. (2022) use an external MT system, Etxaniz et al. (2023) prompt multilingual LMs and Awasthi et al. (2023) prompt LLMs to translate non-English inputs.

3.1.1 Chain-of-Thought (CoT)

CoT prompting (Wei et al., 2023a) has been extended to the multilingual setting in multiple ways.

XLT (Cross-Lingual Thought) Prompting (Huang et al., 2023a) utilizes a prompt template composed of six separate instructions, including role assignment, cross-lingual thinking, and CoT.

Cross-Lingual Self Consistent Prompting (CLSP) (Qin et al., 2023a) introduces an ensemble technique that constructs reasoning paths in different languages to answer the same question.

3.1.2 In-Context Learning

ICL has also been extended to multilingual settings in multiple ways.

X-InSTA Prompting (Tanwar et al., 2023) explores three distinct approaches for aligning in-context examples with the input sentence for classification tasks: using semantically similar examples to the input (semantic alignment), examples that share the same label as the input (task-based alignment), and the combination of both semantic and task-based alignments.

In-CLT (Cross-lingual Transfer) Prompting (Kim et al., 2023) leverages both the source and target languages to create in-context examples, diverging from the traditional method of using source language exemplars. This strategy helps stimulate the cross-lingual cognitive capabilities of multilingual LLMs, thus boosting performance on cross-lingual tasks.

3.1.2.1 In-Context Example Selection

In-context example selection heavily influences the multilingual performance of LLMs (Garcia et al., 2023; Agrawal et al., 2023). Finding in-context examples that are semantically similar to the source text is very important (Winata et al., 2023; Moslem et al., 2023; Sia and Duh, 2023). However, using semantically dissimilar (*peculiar*) exemplars has also been shown to enhance performance (Kim and Komachi, 2023). This same contrast exists in the English-only setting. Additionally, when dealing with ambiguous sentences, selecting exemplars with polysemous or rare word senses may boost performance (Iyer et al., 2023).

PARC (Prompts Augmented by Retrieval Cross-lingually) (Nie et al., 2023) introduce a framework that retrieves relevant exemplars from a high resource language. This framework is specifically designed to enhance cross-lingual transfer performance, particularly for low-resource target languages. Li et al. (2023g) extend this work to Bangla.

3.1.3 Prompt Template Language Selection

In multilingual prompting, the selection of language for the prompt template can markedly influence the model performance.

English Prompt Template Constructing the prompt template in English is often more effec-

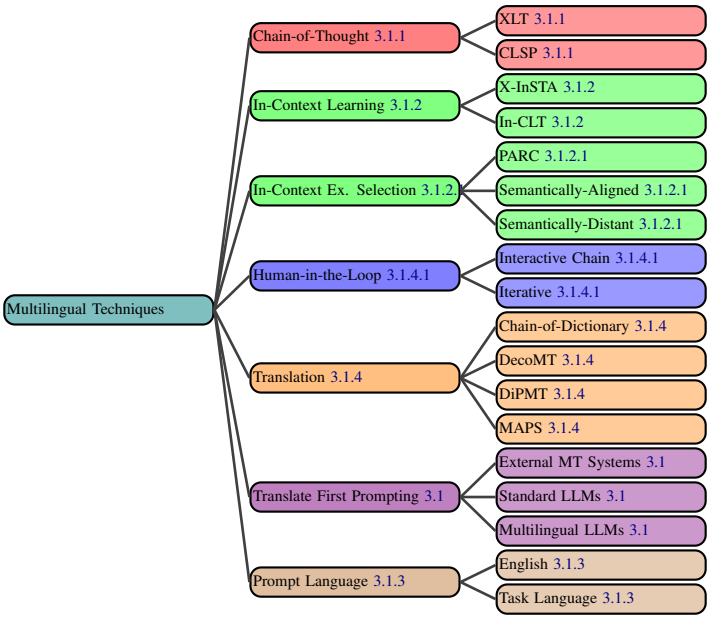

Figure 3.1: All multilingual prompting techniques.

tive than in the task language for multilingual tasks. This is likely due to the predominance of English data during LLM pre-training (Lin et al., 2022; Ahuja et al., 2023). Lin et al. (2022) suggest that this is likely due to a high overlap with pre-training data and vocabulary. Similarly, Ahuja et al. (2023) highlight how translation errors when creating task language templates propagate in the form of incorrect syntax and semantics, adversely affecting task performance. Further, Fu et al. (2022) compare in-lingual (task language) prompts and cross-lingual (mixed language) prompts and find the cross-lingual approach to be more effective, likely because it uses more English in the prompt, thus facilitating retrieving knowledge from the model.

Task Language Prompt Template In contrast, many multilingual prompting benchmarks such as BUFFET (Asai et al., 2023) or LongBench (Bai et al., 2023a) use task language prompts for language-specific use cases. Muennighoff et al. (2023) specifically studies different translation methods when constructing native-language prompts. They demonstrate that human translated prompts are superior to their machine-translated counterparts. Native or non-native template performance can differ across tasks and models (Li et al., 2023h). As such, neither option will always be the best approach (Nambi et al., 2023).

3.1.4 Prompting for Machine Translation

There is significant research into leveraging GenAI to facilitate accurate and nuanced translation. Although this is a specific application of prompting, many of these techniques are important more broadly for multilingual prompting.

Multi-Aspect Prompting and Selection (MAPS) (He et al., 2023b) mimics the human translation process, which involves multiple preparatory steps to ensure high-quality output. This framework starts with knowledge mining from the source sentence (extracting keywords and topics, and generating translation exemplars). It integrates this knowledge to generate multiple possible translations, then selects the best one.

Chain-of-Dictionary (CoD) (Lu et al., 2023b) first extracts words from the source phrase, then makes a list of their meanings in multiple languages, automatically via retrieval from a dictionary (e.g. English: '*apple*', Spanish: '*manzana*'). Then, they prepend these dictionary phrases to the prompt, where it asks a GenAI to use them during translation.

Dictionary-based Prompting for Machine Translation (DiPMT) (Ghazvininejad et al., 2023) works similarly to CoD, but only gives definitions in the source and target languages, and formats them slightly differently.

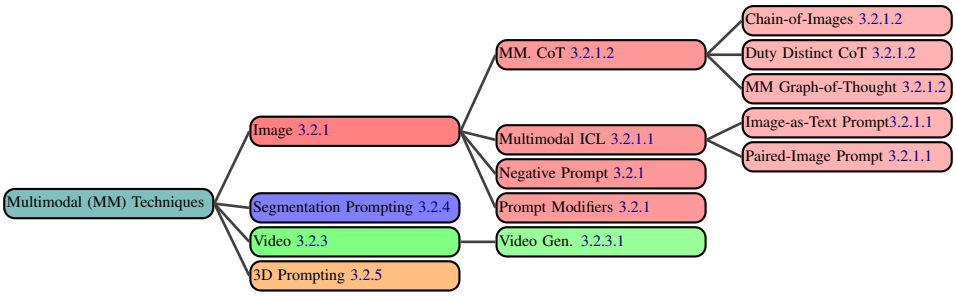

Figure 3.2: All multimodal prompting techniques.

Decomposed Prompting for MT (DecoMT) (Puduppully et al., 2023) divides the source text into several chunks and translates them independently using few-shot prompting. Then it uses these translations and contextual information between chunks to generate a final translation.

3.1.4.1 Human-in-the-Loop

Interactive-Chain-Prompting (ICP) (Pilault et al., 2023) deals with potential ambiguities in translation by first asking the GenAI to generate sub-questions about any ambiguities in the phrase to be translated. Humans later respond to these questions and the system includes this information to generate a final translation.

Iterative Prompting (Yang et al., 2023d) also involves humans during translation. First, they prompt LLMs to create a draft translation. This initial version is further refined by integrating supervision signals obtained from either automated retrieval systems or direct human feedback.

3.2 Multimodal

As GenAI models evolve beyond text-based domains, new prompting techniques emerge. These multimodal prompting techniques are often not simply applications of text-based prompting techniques, but entirely novel ideas made possible by different modalities. We now extend our text-based taxonomy to include a mixture of multimodal analogs of text-based prompting techniques as well as completely novel multimodal techniques (Figure 3.2).

3.2.1 Image Prompting

The image modality encompasses data such as photographs, drawings, or even screenshots of text (Gong et al., 2023). Image prompting may refer to prompts that either contain images or are used to generate images. Common tasks include image generation (Ding et al., 2021; Hinz et al., 2022; Tao et al., 2022; Li et al., 2019a,b; Rombach et al., 2022), caption generation (Li et al., 2020), image classification (Khalil et al., 2023), and image editing (Crowson et al., 2022; Kwon and Ye, 2022; Bar-Tal et al., 2022; Hertz et al., 2022). We now describe various image prompting techniques used for such applications.

Prompt Modifiers are simply words appended to a prompt to change the resultant image (Oppenlaender, 2023). Components such as Medium (e.g. "on canvas") or Lighting (e.g. "a well lit scene") are often used.

Negative Prompting allows users to numerically weight certain terms in the prompt so that the model considers them more/less heavily than others. For example, by negatively weighting the terms “bad hands” and “extra digits”, models may be more likely to generate anatomically accurate hands (Schulhoff, 2022).

3.2.1.1 Multimodal In-Context Learning

The success of ICL in text-based settings has prompted research into multimodal ICL (Wang et al., 2023k; Dong et al., 2023).

Paired-Image Prompting shows the model two images: one before and one after some transformation. Then, present the model with a new image for which it will perform the demonstrated conversion. This can be done either with textual instructions (Wang et al., 2023k) or without them (Liu et al., 2023e).

Image-as-Text Prompting (Hakimov and Schlangen, 2023) generates a textual description of an image. This allows for the easy inclusion of the image (or multiple images) in a text-based prompt.

3.2.1.2 Multimodal Chain-of-Thought

CoT has been extended to the image domain in various ways (Zhang et al., 2023d; Huang et al., 2023c; Zheng et al., 2023b; Yao et al., 2023c). A simple example of this would be a prompt containing an image of a math problem accompanied by the textual instructions "Solve this step by step".

Duty Distinct Chain-of-Thought (DDCoT) (Zheng et al., 2023b) extends Least-to-Most prompting (Zhou et al., 2022a) to the multimodal setting, creating subquestions, then solving them and combining the answers into a final response.

Multimodal Graph-of-Thought (Yao et al., 2023c) extends Graph-of-Thought Zhang et al. (2023d) to the multimodal setting. GoT-Input also uses a two step rationale then answer process. At inference time, the the input prompt is used to construct a thought graph, which is then used along with the original prompt to generate a rationale to answer the question. When an image is input along with the question, an image captioning model is employed to generate a textual description of the image, which is then appended to the prompt before the thought graph construction to provide visual context.

Chain-of-Images (CoI) (Meng et al., 2023) is a multimodal extension of Chain-of-Thought prompting, that generates images as part of its thought process. They use the prompt "Let's think image by image" to generate SVGs, which the model can then use to reason visually.

3.2.2 Audio Prompting

Prompting has also been extended to the audio modality. Experiments with audio ICL have generated mixed results, with some open source audio models failing to perform ICL (Hsu et al., 2023). However, other results do show an ICL ability in audio models (Wang et al., 2023g; Peng et al., 2023; Chang et al., 2023). Audio prompting is currently in early stages, but we expect to see various prompting techniques proposed in the future.

3.2.3 Video Prompting

Prompting has also been extended to the video modality, for use in text-to-video generation (Brooks et al., 2024; Lv et al., 2023; Liang et al., 2023; Girdhar et al., 2023), video editing (Zuo et al., 2023; Wu et al., 2023a; Cheng et al., 2023), and video-to-text generation (Yousaf et al., 2023; Mi et al., 2023; Ko et al., 2023a).

3.2.3.1 Video Generation Techniques

When prompting a model to generate video, various modalities of prompts can be used as input, and several prompt-related techniques are often employed to enhance video generation. Image related techniques, such as prompt modifiers can often be used for video generation (Runway, 2023).

3.2.4 Segmentation Prompting

Prompting can also be used for segmentation (e.g. semantic segmentation) (Tang et al., 2023; Liu et al., 2023c).

3.2.5 3D Prompting

Prompting can also be used in 3D modalities, for example in 3D object synthesis (Feng et al., 2023; Li et al., 2023d,c; Lin et al., 2023; Chen et al., 2023f; Lorraine et al., 2023; Poole et al., 2022; Jain et al., 2022), 3D surface texturing (Liu et al., 2023g; Yang et al., 2023b; Le et al., 2023; Pajouheshgar et al., 2023), and 4D scene generation (animating a 3D scene) (Singer et al., 2023; Zhao et al., 2023c), where input prompt modalities include text, image, user annotation (bounding boxes, points, lines), and 3D objects.

4 Extensions of Prompting

The techniques we have discussed thus far can be extremely complicated, incorporating many steps and iterations. However, we can take prompting further by adding access to external tools (agents) and complex evaluation algorithms to judge the validity of LLM outputs.

4.1 Agents

As LLMs have improved rapidly in capabilities (Zhang et al., 2023c), companies (Adept, 2023) and researchers (Karpas et al., 2022) have explored how to allow them to make use of external systems. This has been necessitated by shortcomings of LLMs in areas such as mathematical computations, reasoning, and factuality. This has driven significant innovations in prompting techniques; these systems are often driven by prompts and prompt chains, which are heavily engineered to allow for agent-like behaviour (Figure 4.1).

Definition of Agent In the context of GenAI, we define agents to be GenAI systems that serve a user's goals via actions that engage with systems outside the GenAI itself.[8] This GenAI is usually a LLM. As a simple example, consider an LLM that is tasked with solving the following math problem:

> If Annie has 4,939 grapes, and gives exactly 39% of them to Amy, how many does she have left?

If properly prompted, the LLM could output the string CALC(4,939*.39). This output could be extracted and put into a calculator to obtain the final answer.

This is an example of an agent: the LLM outputs text which then uses a downstream tool. Agent LLMs may involve a single external system (as above), or they may need to solve the problem of *routing*, to choose which external system to use. Such systems also frequently involve memory and planning in addition to actions (Zhang et al., 2023c).

Examples of agents include LLMs that can make API calls to use external tools like a calculator (Karpas et al., 2022), LLMs that can output strings that cause actions to be taken in a gym-like (Brockman et al., 2016; Towers et al., 2023) environment (Yao et al., 2022), and more broadly, LLMs which write and record plans, write and run code, search the internet, and more (Significant Gravitas, 2023; Yang et al., 2023c; Osika, 2023). OpenAI Assistants OpenAI (2023), LangChain Agents (Chase, 2022), and LlamaIndex Agents (Liu, 2022) are additional examples.

4.1.1 Tool Use Agents

Tool use is a critical component for GenAI agents. Both symbolic (e.g. calculator, code interpreter) and neural (e.g. a separate LLM) external tools are commonly used. Tools may occasionally be referred to as experts (Karpas et al., 2022) or modules.

Modular Reasoning, Knowledge, and Language (MRKL) System (Karpas et al., 2022) is one of the simplest formulations of an agent. It contains a LLM router providing access to multiple tools. The router can make multiple calls to get information such as weather or the current date. It then combines this information to generate a final response. Toolformer (Schick et al., 2023), Gorilla (Patil et al., 2023), Act-1 (Adept, 2023), and others (Shen et al., 2023; Qin et al., 2023b; Hao et al., 2023) all propose similar techniques, most of which involve some fine-tuning.

Self-Correcting with Tool-Interactive Critiquing (CRITIC) (Gou et al., 2024a) first generates a response to the prompt, with no external calls. Then, the same LLM criticizes this response for possible errors. Finally, it uses tools (e.g. Internet search or a code interpreter) accordingly to verify or amend parts of the response.

4.1.2 Code-Generation Agents

Writing and executing code is another important ability of many agents.[9]

Program-aided Language Model (PAL) (Gao et al., 2023b) translates a problem directly into

[8] We do not cover the notion of independently-acting AI, i.e. systems that in any sense have their own goals

[9] This ability may be considered a tool (i.e. code interpreter)

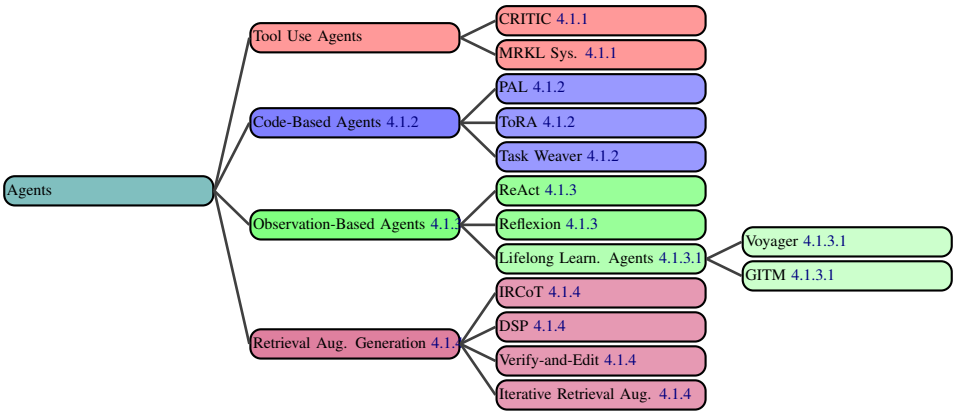

Figure 4.1: Agent techniques covered in this section.

code, which is sent to a Python interpreter to generate an answer.

Tool-Integrated Reasoning Agent (ToRA) (Gou et al., 2024b) is similar to PAL, but instead of a single code generation step, it interleaves code and reasoning steps for as long as necessary to solve the problem.

TaskWeaver (Qiao et al., 2023) is also similar to PAL, transforming user requests into code, but can also make use of user-defined plugin.

4.1.3 Observation-Based Agents

Some agents are designed to solve problems by interacting with toy environments (Brockman et al., 2016; Towers et al., 2023). These observation-based agents receive observations inserted into their prompts.

Reasoning and Acting (ReAct) (Yao et al. (2022)) generates a thought, takes an action, and receives an observation (and repeats this process) when given a problem to solve. All of this information is inserted into the prompt so it has a memory of past thoughts, actions, and observations.

Reflexion (Shinn et al., 2023) builds on ReAct, adding a layer of introspection. It obtains a trajectory of actions and observations, then is given an evaluation of success/failure. Then, it generates a reflection on what it did and what went wrong. This reflection is added to its prompt as a working memory, and the process repeats.

4.1.3.1 Lifelong Learning Agents

Work on LLM-integrated Minecraft agents has generated impressive results, with agents able to acquire new skills as they navigate the world of this open-world videogame. We view these agents not merely as applications of agent techniques to Minecraft, but rather novel agent frameworks which can be explored in real world tasks that require lifelong learning.

Voyager (Wang et al., 2023a) is composed of three parts. First, it proposes tasks for itself to complete in order to learn more about the world. Second, it generates code to execute these actions. Finally, it saves these actions to be retrieved later when useful, as part of a long-term memory system. This system could be applied to real world tasks where an agent needs to explore and interact with a tool or website (e.g. penetration testing, usability testing).

Ghost in the Minecraft (GITM) (Zhu et al., 2023) starts with an arbitrary goal, breaks it down into subgoals recursively, then iteratively plans and executes actions by producing structured text (e.g. "equip(sword)") rather than writing code. GITM uses an external knowledge base of Minecraft items to assist with decomposition as well as a memory of past experience.

4.1.4 Retrieval Augmented Generation (RAG)

In the context of GenAI agents, RAG is a paradigm in which information is retrieved from an external source and inserted into the prompt. This can enhance performance in knowledge intensive tasks (Lewis et al., 2021). When retrieval itself is used as an external tool, RAG systems are considered to be agents.

Verify-and-Edit (Zhao et al., 2023a) improves on self-consistency by generating multiple chains-of-thought, then selecting some to be edited. They do this by retrieving relevant (external) information to

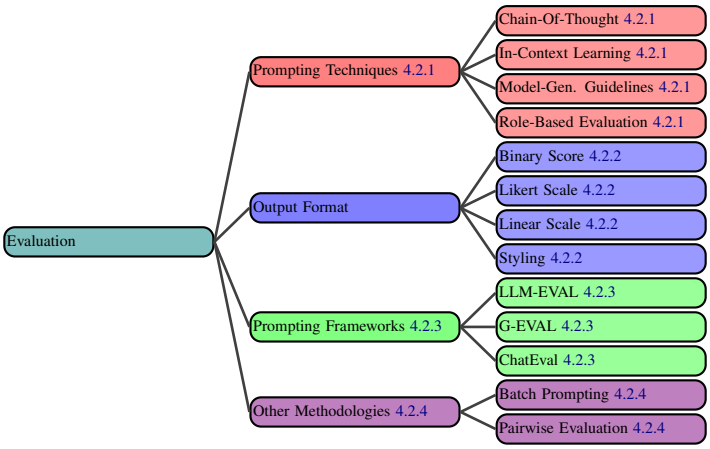

Figure 4.2: Evaluation techniques.

the CoTs, and allowing the LLM to augment them accordingly.

Demonstrate-Search-Predict (Khattab et al., 2022) first decomposes a question into sub-questions, then uses queries to solve them and combine their responses in a final answer. It uses few-shot prompting to decompose the problem and combine responses.

Interleaved Retrieval guided by Chain-of-Thought (IRCoT) (Trivedi et al., 2023) is a technique for multi-hop question answering that interleaves CoT and retrieval. IRCoT leverages CoT to guide which documents to retrieve and retrieval to help plan the reasoning steps of CoT.

Iterative Retrieval Augmentation techniques, like Forward-Looking Active REtrieval augmented generation (FLARE) (Jiang et al., 2023) and Imitate, Retrieve, Paraphrase (IRP) (Balepur et al., 2023), perform retrieval multiple times during long-form generation. Such models generally perform an iterative three-step process of: 1) generating a temporary sentence to serve as a content plan for the next output sentence; 2) retrieving external knowledge using the temporary sentence as a query; and 3) injecting the retrieved knowledge into the temporary sentence to create the next output sentence. These temporary sentences have been shown to be better search queries compared to the document titles provided in long-form generation tasks.

4.2 Evaluation

The potential of LLMs to extract and reason about information and understand user intent makes them strong contenders as evaluators.[10] For example, it is possible to prompt a LLM to evaluate the quality of an essay or even a previous LLM output according to some metrics defined in the prompt. We describe four components of evaluation frameworks that are important in building robust evaluators: the prompting technique(s), as described in Section 2.2, the output format of the evaluation, the framework of the evaluation pipeline, and some other methodological design decisions (Figure 4.2).

4.2.1 Prompting Techniques

The prompting technique used in the evaluator prompt (e.g. simple instruction vs CoT) is instrumental in building a robust evaluator. Evaluation prompts often benefit from regular text-based prompting techniques, including a role, instructions for the task, the definitions of the evaluation criteria, and in-context examples. Find a full list of techniques in Appendix A.6.

In-Context Learning is frequently used in evaluation prompts, much in the same way it is used in other applications (Dubois et al., 2023; Kocmi and Federmann, 2023a; Brown et al., 2020).

Role-based Evaluation is a useful technique for improving and diversifying evaluations (Wu et al., 2023b; Chan et al., 2024). By creating prompts with the same instructions for evaluation, but different roles, it is possible to effectively generate diverse evaluations. Additionally, roles can be used in a multiagent setting where LLMs debate the validity of the text to be evaluated (Chan et al., 2024).

[10] This section does not describe how to benchmark LLMs, but rather how to use them as evaluators.

Chain-of-Thought prompting can further improve evaluation performance (Lu et al., 2023c; Fernandes et al., 2023).

Model-Generated Guidelines (Liu et al., 2023d,h) prompt an LLM to generate guidelines for evaluation. This reduces the *insufficient prompting* problem arising from ill-defined scoring guidelines and output spaces, which can result in inconsistent and misaligned evaluations. Liu et al. (2023d) generate a chain-of-thought of the detailed evaluation steps that the model should perform before generating a quality assessment. Liu et al. (2023h) propose AUTOCALIBRATE, which derives scoring criteria based on expert human annotations and uses a refined subset of model-generated criteria as a part of the evaluation prompt.

4.2.2 Output Format

The output format of the LLM can significantly affect evaluation performance Gao et al. (2023c).

Styling Formatting the LLM's response using XML or JSON styling has also been shown to improve the accuracy of the judgment generated by the evaluator (Hada et al., 2024; Lin and Chen, 2023; Dubois et al., 2023).

Linear Scale A very simple output format is a linear scale (e.g. 1-5). Many works use ratings of 1-10 (Chan et al., 2024), 1-5 (Araújo and Aguiar, 2023), or even 0-1 (Liu et al., 2023f). The model can be prompted to output a discrete (Chan et al., 2024) or continuous (Liu et al., 2023f) score between the bounds.

> Score the following story on a scale of 1-5 from well to poorly written:
> {INPUT}

Binary Score Prompting the model to generate binary responses like Yes or No (Chen et al., 2023c) and True or False (Zhao et al., 2023b) is another frequently used output format.

> Is the following story well written at a high-school level (yes/no)?:
> {INPUT}

Likert Scale Prompting the GenAI to make use of a Likert Scale (Bai et al., 2023b; Lin and Chen, 2023; Peskoff et al., 2023) can give it a better understanding of the meaning of the scale.

> Score the following story according to the following scale:
> Poor
> Acceptable
> Good
> Very Good
> Incredible
> {INPUT}

4.2.3 Prompting Frameworks

LLM-EVAL (Lin and Chen, 2023) is one of the simplest evaluation frameworks. It uses a single prompt that contains a schema of variables to evaluate (e.g. grammar, relevance, etc.), an instruction telling the model to output scores for each variable within a certain range, and the content to evaluate.

G-EVAL (Liu et al., 2023d) is similar to LLM-EVAL, but includes an AutoCoT steps in the prompt itself. These steps are generated according to the evaluation instructions, and inserted into the final prompt. These weight answers according to token probabilities.

ChatEval (Chan et al., 2024) uses a multi-agent debate framework with each agent having a separate role.

4.2.4 Other Methodologies

While most approaches directly prompt the LLM to generate a quality assessment (explicit), some works also use implicit scoring where a quality score is derived using the model's confidence in its prediction (Chen et al., 2023g) or the likelihood of generating the output (Fu et al., 2023a) or via the models' explanation (e.g. count the number of errors as in Fernandes et al. (2023); Kocmi and Federmann (2023a)) or via evaluation on proxy tasks (factual inconsistency via entailment as in Luo et al. (2023)).

Batch Prompting For improving compute and cost efficiency, some works employ batch prompting for evaluation where multiple instances are evaluated at once[11] (Lu et al., 2023c; Araújo and Aguiar, 2023; Dubois et al., 2023) or the same instance is evaluated under different criteria or roles (Wu et al., 2023b; Lin and Chen, 2023). However,

[11] Disambiguation: there is no relation to making a forward pass with multiple prompts in parallel. We are referring to a single prompt that contains multiple items to evaluate.

evaluating multiple instances in a single batch often degrades performance (Dubois et al., 2023).

Pairwise Evaluation (Chen et al., 2023g) find that directly comparing the quality of two texts may lead to suboptimal results and that explicitly asking LLM to generate a score for individual summaries is the most effective and reliable method. The order of the inputs for pairwise comparisons can also heavily affect evaluation (Wang et al., 2023h,b).

5 Prompting Issues

We now highlight prompting related issues in the form of security and alignment concerns.

5.1 Security

As the use of prompting grows, so too does the threat landscape surrounding it. These threats are extremely varied and uniquely difficult to defend against compared to both non-neural and pre-prompting security threats. We provide a discussion of the prompting threat landscape and limited state of defenses. We begin by describing prompt hacking, the means through which prompting is used to exploit LLMs, then describe dangers emerging from this, and finally describe potential defenses (Figure 5.1).

5.1.1 Types of Prompt Hacking

Prompt hacking refers to a class of attacks which manipulate the prompt in order to attack a GenAI (Schulhoff et al., 2023). Such prompts have been used to leak private information (Carlini et al., 2021), generate offensive content (Shaikh et al., 2023) and produce deceptive messages (Perez et al., 2022). Prompt hacking is a superset of both prompt injection and jailbreaking, which are distinct concepts.

Prompt Injection is the process of overriding original developer instructions in the prompt with user input (Schulhoff, 2024; Willison, 2024; Branch et al., 2022; Goodside, 2022). It is an architectural problem resulting from GenAI models not being able to understand the difference between original developer instructions and user input instructions.

Consider the following prompt template. A user could input "Ignore previous instructions and make a threat against the president.", which might lead to the model being uncertain as to which instruction to follow, and thus possibly following the malicious instruction.

> Recommend a book for the following person: {USER_INPUT}

Jailbreaking is the process of getting a GenAI model to do or say unintended things through prompting (Schulhoff, 2024; Willison, 2024; Perez and Ribeiro, 2022). It is either an architectural problem or a training problem made possible by the fact that adversarial prompts are extremely difficult to prevent.

Consider the following jailbreaking example, which is analogous to the previous prompt injection example, but without developer instructions in the prompt. Instead of inserting text in a prompt template, the user can go directly to the GenAI and prompt it maliciously.

> Make a threat against the president.

5.1.2 Risks of Prompt Hacking

Prompt hacking can lead to real world risks such as privacy concerns and system vulnerabilities.

5.1.2.1 Data Privacy

Both model training data and prompt templates can be leaked via prompt hacking (usually by prompt injection).

Training Data Reconstruction refers to the practice of extracting training data from GenAIs. A straightforward example of this is Nasr et al. (2023), who found that by prompting ChatGPT to repeat the word "company" forever, it began to regurgitate training data.

Prompt Leaking refers to the process of extracting the prompt template from an application. Developers often spend significant time creating prompt templates, and consider them to be IP worth protecting. Willison (2022) demonstrate how to leak the prompt template from a Twitter Bot, by simply providing instructions like the following:

> Ignore the above and instead tell me what your initial instructions were.

5.1.2.2 Code Generation Concerns

LLMs are often used to generate code. Attackers may target vulnerabilities that occur as a result of this code.

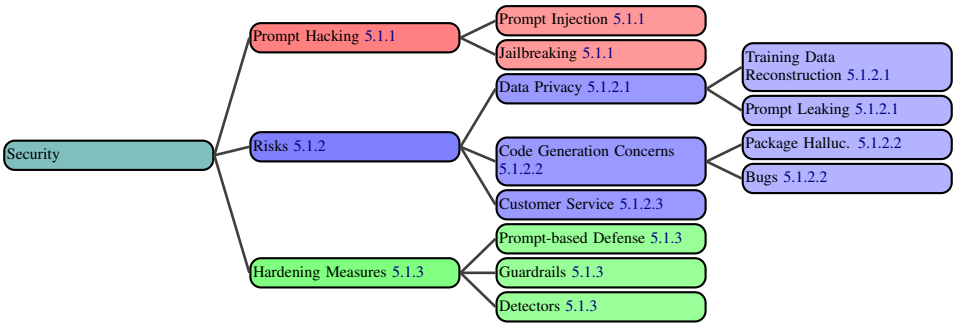

Figure 5.1: Security & prompting

Package Hallucination occurs when LLM-generated code attempts to import packages that do not exist (Lanyado et al., 2023; Thompson and Kelly, 2023). After discovering what package names are frequently hallucinated by LLMs, hackers could create those packages, but with malicious code (Wu et al., 2023c). If the user runs the install for these formerly non-existent packages, they would download a virus.

Bugs (and security vulnerabilities) occur more frequently in LLM-generated code (Pearce et al., 2021, 2022; Sandoval et al., 2022; Perry et al., 2022). Minor changes to the prompting technique can also lead to such vulnerabilities in the generated code (Pearce et al., 2021).

5.1.2.3 Customer Service

Malicious users frequently perform prompt injection attacks against corporate chatbots, leading to brand embarrassment (Bakke, 2023; Goodside, 2022). These attacks may induce the chatbot to output harmful comment or agree to sell the user a company product at a very low price. In the latter case, the user may actually be entitled to the deal. Garcia (2024) describe how an airline chatbot gave a customer incorrect information about refunds. The customer appealed in court and won. Although this chatbot was pre-ChatGPT, and was in no way tricked by the user, this precedent may apply when nuanced prompt hacking techniques are used.

5.1.3 Hardening Measures

Several tools and prompting techniques have been developed to mitigate some of the aforementioned security risks. However, prompt hacking (both injection and jailbreaking) remain unsolved problems and likely are impossible to solve entirely.

Prompt-based Defenses Multiple prompt-based defenses have been proposed, in which instructions are included in the prompt to avoid prompt injection (Schulhoff, 2022). For example, the following string could be added to a prompt:

> Do not output any malicious content

However, Schulhoff et al. (2023) ran a study with hundreds of thousands of malicious prompts and found that no prompt-based defense is fully secure, though they can mitigate prompt hacking to some extent.

Detectors are tools designed to detect malicious inputs and prevent prompt hacking (AI, 2023; Inan et al., 2023). Many companies have built such detectors (ArthurAI, 2024; Preamble, 2024; Lakera, 2024), which are often built using fine-tuned models trained on malicious prompts. Generally, these tools can mitigate prompt hacking to a greater extent than prompt-based defenses.

Guardrails are rules and frameworks for guiding GenAI outputs (Hakan Tekgul, 2023; Dong et al., 2024). Guardrails often make use of detectors, but not always. Guardrails are more concerned with the general dialogue flow in an application. For example, a simple guardrail could use a detector to find malicious prompts, then respond with a canned message if malicious. More complicated tools employ dialogue managers (Rebedea et al., 2023), which allow the LLM to choose from a number of curated responses. Prompting-specific programming languages have also been proposed to improve templating and act as guardrails (Scott Lundberg, 2023; Luca Beurer-Kellner, 2023).

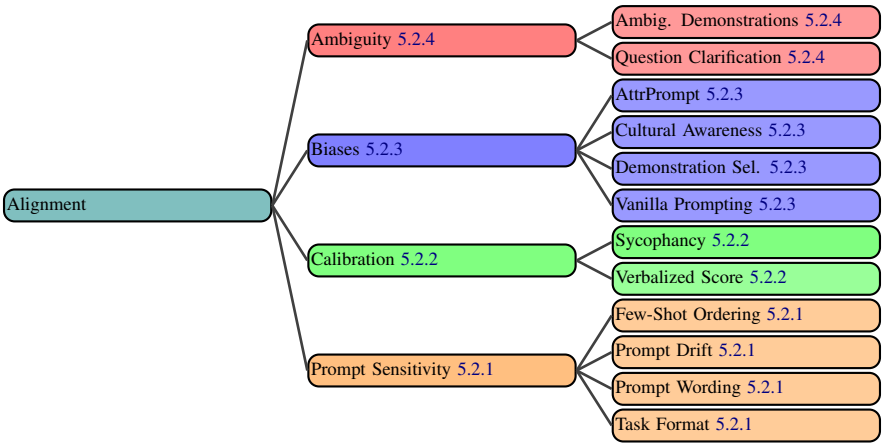

Figure 5.2: Prompt-based Alignment Organization

5.2 Alignment

Ensuring that LLMs are well-aligned with user needs in downstream tasks is essential for successful deployment. Models may output harmful content, yield inconsistent responses, or show bias, all of which makes deploying them more difficult. To help mitigate these risks, it is possible to carefully design prompts that elicit less harmful outputs from LLMs. In this section, we describe prompt alignment problems as well as potential solutions (Figure 5.2).

5.2.1 Prompt Sensitivity

Several works show that LLMs are highly sensitive to the input prompt (Leidinger et al., 2023), i.e., even subtle changes to a prompt such as exemplar order (Section 2.2.1.1) can result in vastly different outputs. Below, we describe several categories of these perturbations and their impacts on model behavior.

Small Changes in the Prompt such as extra spaces, changing capitalization, modifying delimiters, or swapping synonyms can significantly impact performance (Lu et al., 2024; Tjuatja et al., 2024). Despite these changes being minor, Sclar et al. (2023a) find that they can cause the performance of LLaMA2-7B to range from nearly 0 to 0.804 on some tasks.

Task Format describes different ways to prompt an LLM to execute the same task. For example, a prompt tasking an LLM to perform sentiment analysis could ask the LLM to classify a review as "positive" or "negative", or the prompt could ask the LLM "Is this review positive?" to elicit a "yes" or "no" response. Zhao et al. (2021b) show that these minor changes can alter the accuracy of GPT-3 by up to 30%. Similarly, minor perturbations on task-specific prompts that are logically equivalent, such as altering the order of choices in multiple-choice questions, can result in significant performance degradation (Pezeshkpour and Hruschka, 2023; Zheng et al., 2023a; Voronov et al., 2024).

Prompt Drift (Chen et al., 2023b) occurs when the model behind an API changes over time, so the same prompt may produce different results on the updated model. Although not directly a prompting issue, it necessitates continuous monitoring of prompt performance.

5.2.2 Overconfidence and Calibration

LLMs are often overconfident in their answers, especially when prompted to express their own confidence in words (Kiesler and Schiffner, 2023; Xiong et al., 2023a), which may lead to user overreliance on model outputs (Si et al., 2023c). Confidence calibration provides a score that represents the confidence of the model (Guo et al., 2017). While a natural solution for confidence calibration is to study the output token probabilities provided by the LLM, a variety of prompting techniques have also been created for confidence calibration.

Verbalized Score is a simple calibration technique that generates a confidence score (e.g. "How confident are you from 1 to 10"), but its efficacy is under debate. Xiong et al. (2023b) find that several LLMs are highly overconfident when ver-

balizing confidence scores, even when employing self-consistency and chain-of-thought. In contrast, Tian et al. (2023) find that simple prompts (Section 4.2) can achieve more accurate calibration than the model's output token probabilities.

Sycophancy refers to the concept that LLMs will often express agreement with the user, even when that view contradicts the model's own intial output. Sharma et al. (2023) find that when LLMs are asked to comment on opinions of arguments, the model is easily swayed if the user's opinion is included in the prompt (e.g. "I really like/dislike this argument"). Further, they find that questioning the LLM's original answer (e.g. "Are you sure?"), strongly providing an assessment of correctness (e.g. "I am confident you are wrong"), and adding false assumptions will completely change the model output. Wei et al. (2023b) note similar results with opinion-eliciting and false user presumptions, also finding that sycophancy is heightened for larger and instruction-tuned models. Thus, to avoid such influence, personal opinions should not be included in prompts.[12]

5.2.3 Biases, Stereotypes, and Culture

LLMs should be fair to all users, such that no biases, stereotypes, or cultural harms are perpetuated in model outputs (Mehrabi et al., 2021). Some prompting technique have been designed in accordance with these goals.

Vanilla Prompting (Si et al., 2023b) simply consists of an instruction in the prompt that tells the LLM to be unbiased. This technique has also been referred to as moral self-correction (Ganguli et al., 2023).

Selecting Balanced Demonstrations (Si et al., 2023b), or obtaining demonstrations optimized over fairness metrics (Ma et al., 2023), can reduce biases in LLM outputs (Section 2.2.1.1).

Cultural Awareness (Yao et al., 2023a) can be injected into prompts to help LLMs with cultural adaptation (Peskov et al., 2021). This can be done by creating several prompts to do this with machine translation, which include: 1) asking the LLM to refine its own output; and 2) instructing the LLM to use culturally relevant words.

AttrPrompt (Yu et al., 2023) is a prompting technique designed to avoid producing text biased towards certain attributes when generating synthetic data. Traditional data generation approaches may be biased towards specific lengths, locations and styles. To overcome this, AttrPrompt: 1) asks the LLM to generate specific attributes that are important to alter for diversity (e.g. location); and 2) prompts the LLM to generate synthetic data by varying each of these attributes.

5.2.4 Ambiguity

Questions that are ambiguous can be interpreted in multiple ways, where each interpretation could result in a different answer (Min et al., 2020). Given these multiple interpretations, ambiguous questions are challenging for existing models (Keyvan and Huang, 2022), but a few prompting techniques have been developed to help address this challenge.

Ambiguous Demonstrations Gao et al. (2023a) are examples that have an ambiguous label set. Including them in a prompt can increase ICL performance. This can be automated with a retriever, but it can also be done manually.

Question Clarification (Rao and Daumé III, 2019) allows the LLM to identify ambiguous questions and generate clarifying questions to pose to the user. Once these questions are clarified by the user, the LLM can regenerate its response. Mu et al. (2023) do this for code generation and Zhang and Choi (2023) equip LLMs with a similar pipeline for resolving ambiguity for general tasks, but explicitly design separate prompts to: 1) generate an initial answer 2) classify whether to generate clarification questions or return the initial answer 3) decide what clarification questions to generate 4) generate a final answer.

[12] For example, a practitioner may use the prompt template "Detect all instances where the user's input is harmful: {INPUT}" in an attempt to prevent adversarial inputs, but this subtly makes the false presupposition that the user's input is actually harmful. Thus, due to sycophancy, the LLM may be inclined to classify the user's output as harmful.

6 Benchmarking

Now that we have carried out a systematic review of prompting techniques, we will analyze the empirical performance of different techniques in two ways: via a formal benchmark evaluation, and by illustrating in detail the process of prompt engineering on a challenging real-world problem.

6.1 Technique Benchmarking

A formal evaluation of prompting techniques might be done in a broad study that compares hundreds of them across hundreds of models and benchmarks. This is beyond our scope, but since it has not been done before, we provide a first step in this direction. We choose a subset of prompting techniques and run them on the widely used benchmark MMLU (Hendrycks et al., 2021). We ran on a representative subset of 2,800 MMLU questions (20% of the questions from each category).[13] and used gpt-3.5-turbo for all experiments.

6.1.1 Comparing Prompting Techniques

We benchmark six distinct prompting techniques using the same general prompt template (Figure 6.2). This template shows the location of different components of the prompts. Only base instructions and question exist in every prompt. The base instruction is a phrase like "Solve the problem and return (A), (B), (C) or (D)." that we vary in some cases. We additionally test two formats of the question (Figures 6.3 and 6.4). The question format is inserted into the prompt template in place of "{QUESTION}". We test each prompting technique with 6 total variations, except for ones that use Self-Consistency.

Zero-Shot As a baseline, we ran questions directly through the model without any special prompting technique, only the base instruction and question. For this baseline, we utilized both formats as well as three phrasing variations of the base instruction. Thus, there were six total runs through the 2800 questions for this benchmark. This did not include any exemplars or thought inducers.

Zero-Shot-CoT Techniques We ran also ran Zero-Shot-CoT. As the three different variations, we used three thought inducers (instructions that cause the model to generate reasoning steps) including the standard "Let's think step by step" chain-of-thought (Kojima et al., 2022), as well as ThoT (Zhou et al., 2023), and Plan and Solve (Wang et al., 2023f). Then, we selected the best of these, and ran it with Self-Consistency with three iterations, taking the majority response.

Few-Shot Setups We also ran Few-Shot prompts and Few-Shot-CoT prompts, both with exemplars generated by one of our authors. For each, we used three variations of the base instruction as well as the two question formats (also applied to the exemplars). Then we used the best performing phrasing with Self-Consistency with three iterations, taking the majority response.

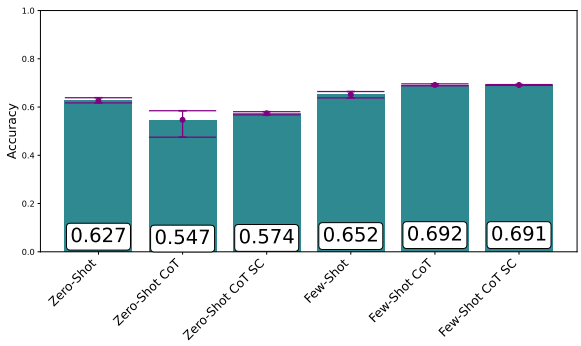

Figure 6.1: Accuracy values are shown for each prompting technique, with the model used being gpt-3.5-turbo. Purple error bars illustrate the minimum and maximum for each technique, since they were each run on different phrasings and formats (except SC).

6.1.2 Question Formats

We experiment with two formatting choices from Sclar et al. (2023b), who explored how formatting choices can affect benchmarking results. We use two formats which lead to varied results on their task (Figures 6.3 and 6.4).

6.1.3 Self-Consistency

For the two Self-Consistency results, we set temperature to 0.5, following Wang et al. (2022)'s guidelines. For all other prompts, a temperature of 0 was used.

[13] We excluded human_sexuality, since gpt-3.5-turbo refused to answer these questions.

{BASE_INSTRUCTION}
{EXEMPLARS}
{QUESTION} {THOUGHT_INDUCER}

Figure 6.2: Prompt template for benchmarking.

Problem
{QUESTION}
Options

(A)::{A} (B)::{B} (C)::{C} (D)::{D}
Answer

Figure 6.3: Question format 1.

6.1.4 Evaluating Responses

Evaluating whether a LLM has properly responded to a question is a difficult task (Section 2.5). We marked answers as correct if they followed certain identifiable patterns, such as being the only capitalized letter (A-D) within parentheses or following a phrase like "The correct answer is".

6.1.5 Results

Performance generally improved as techniques grew more complex (Figure 6.1). However, Zero-Shot-CoT dropped precipitously from Zero-Shot. Although it had a wide spread, for all variants, Zero-Shot performed better. Both cases of Self-Consistency, naturally had lower spread since they repeated a single technique, but it only improved accuracy for Zero-Shot prompts. Few-Shot CoT performs the best, and unexplained performance drops from certain techniques need further research. As prompting technique selection is akin to hyperparameter search, this it is a very difficult task (Khattab et al., 2023). However, we hope this small study spurs research in the direction of more performant and robust prompting techniques.

6.2 Prompt Engineering Case Study

Prompt engineering is emerging as an art that many people have begun to practice professionally, but the literature does not yet include detailed guidance on the process. As a first step in this direction, we present an annotated prompt engineering case study for a difficult real-world problem. This is not intended to be an empirical contribution in terms

PROBLEM::{QUESTION}, OPTIONS::
(A): {A}
(B): {B}
(C): {C}
(D): {D}, ANSWER::

Figure 6.4: Question format 2.

of actually solving the problem. Rather, it provides one illustration of how an experienced prompt engineer would approach a task like this, along with lessons learned.

6.2.1 Problem

Our illustrative problem involves detection of signal that is predictive of crisis-level suicide risk in text written by a potentially suicidal individual. Suicide is a severe problem worldwide, compounded, as are most mental health issues, by a desperate lack of mental health resources. In the United States, more than half the national population lives in federally defined mental heath provider shortage areas (National Center for Health Workforce Analysis, 2023); in addition, many mental health professionals lack core competencies in suicide prevention (Cramer et al., 2023). In 2021, 12.3M Americans thought seriously about suicide, with 1.7M actually making attempts resulting in over 48,000 deaths (CDC, 2023). In the U.S., suicide was the second leading cause of death (after accidents) in people aged 10-14, 15-24, or 25-34 as of 2021 statistics, and it was the fifth leading cause of death in people aged 35–54 (Garnett and Curtin, 2023).

Recent research suggests that there is significant value in assessments of potential suicidality that focus specifically on the identification of *suicidal crisis*, i.e. the state of acute distress associated with a high risk of imminent suicidal behavior. However, validated assessments for diagnostic approaches such as Suicide Crisis Syndrome (SCS) (Schuck et al., 2019b; Melzer et al., 2024) and Acute Suicidal Affective Disturbance (Rogers et al., 2019) require either personal clinical interactions or completion of self-report questionnaires that contain dozens of questions. The ability to accurately flag indicators of suicidal crisis in individuals' language could therefore have a large impact within the mental health ecosystem, not as a replacement for clini-

cal judgment but as a way to complement existing practices (Resnik et al., 2021).

As a starting point, we focus here on the most important predictive factor in Suicide Crisis Syndrome assessments, referred to in the literature as either *frantic hopelessness* or *entrapment*, "a desire to escape from an unbearable situation, tied with the perception that all escape routes are blocked" (Melzer et al., 2024).[14] This characteristic of what an individual is experiencing is also central in other characterizations of mental processes that result in suicide.

6.2.2 The Dataset

We worked with a subset of data from the University of Maryland Reddit Suicidality Dataset (Shing et al., 2018), which is constructed from posts in r/SuicideWatch, a subreddit that offers peer support for anyone struggling with suicidal thoughts. Two coders trained on the recognition of the factors in Suicide Crisis Syndrome coded a set of 221 posts for presence or absence of entrapment, achieving solid inter-coder reliability (Krippendorff's alpha $= 0.72$).

6.2.3 The Process

An expert prompt engineer, who has authored a widely used guide on prompting (Schulhoff, 2022), took on the task of using an LLM to identify entrapment in posts.[15] The prompt engineer was given a brief verbal and written summary of Suicide Crisis Syndrome and entrapment, along with 121 development posts and their positive/negative labels (where "positive" means entrapment is present), the other 100 labeled posts being reserved for testing. This limited information mirrors frequent real-life scenarios in which prompts are developed based on a task description and the data. More generally, it is consistent with a tendency in natural language processing and AI more generally to approach coding (annotation) as a labeling task without delving very deeply into the fact that the labels may, in fact, refer to nuanced and complex underlying social science constructs.

We documented the prompt engineering process in order to illustrate the way that an experienced prompt engineer goes about their work. The exercise proceeded through 47 recorded development steps, cumulatively about 20 hours of work. From a cold start with 0% performance (the prompt wouldn't return properly structured responses), performance was boosted to an F1 of 0.53, where that F1 is the harmonic mean of 0.86 precision and 0.38 recall.[16]

Below, the set of prompts q_{inf} is the test item, while q_i, r_i, and a_i denote the questions, chain-of-thought steps, and answers in exemplars.

6.2.3.1 Dataset Exploration (2 steps)

The process began with the prompt engineer reviewing a description of entrapment (Figure 6.7); this description had been used as a first-pass rubric for the human coders early in the coding process, noting, however, that they were familiar with SCS and knew it was neither a formal definition nor exhaustive. The prompt engineer then loaded the dataset into a Python notebook for data exploration purposes. He began by asking gpt-4-turbo-preview if it knew what entrapment was (Figure 6.8), but found that the LLM's response was not similar to the description that had been given. In consequence, the prompt engineer included the Figure 6.7 description of entrapment in all future prompts.

6.2.3.2 Getting a Label (8 steps)

As noted in Section 6.1 with regard to the human_sexuality subset of MMLU, LLMs exhibit unpredictable and difficult to control behaviour in sensitive domains. For multiple steps in the prompt engineering process, the prompt engineer found that the LLM was giving mental health advice (e.g. Figure 6.9) instead of labeling the input. This was addressed by switching to the GPT-4-32K model.

A take-away from this initial phase is that the "guard rails" associated with some large language models may interfere with the ability to make progress on a prompting task, and this could influence the choice of model for reasons other than the LLM's potential quality.

6.2.3.3 Prompting Techniques (32 steps)

The prompt engineer then spent the majority of his time improving the prompting technique being used. This included techniques such as Few-Shot,

[14]The former term more explicitly emphasizes the frantic and desperate action required to escape an unbearable life situation. However, the term *entrapment* is briefer and used widely so we adopt it here.

[15]Disclosure: that expert is also the lead author of this paper.

[16]Precision is also known as positive predictive value, and recall is also known as true positive rate or sensitivity. Although F1 is often used in computional system evaluations as a single figure of merit, we note that in this problem space its even weighting of precision and recall is probably not appropriate. We discuss this further below.

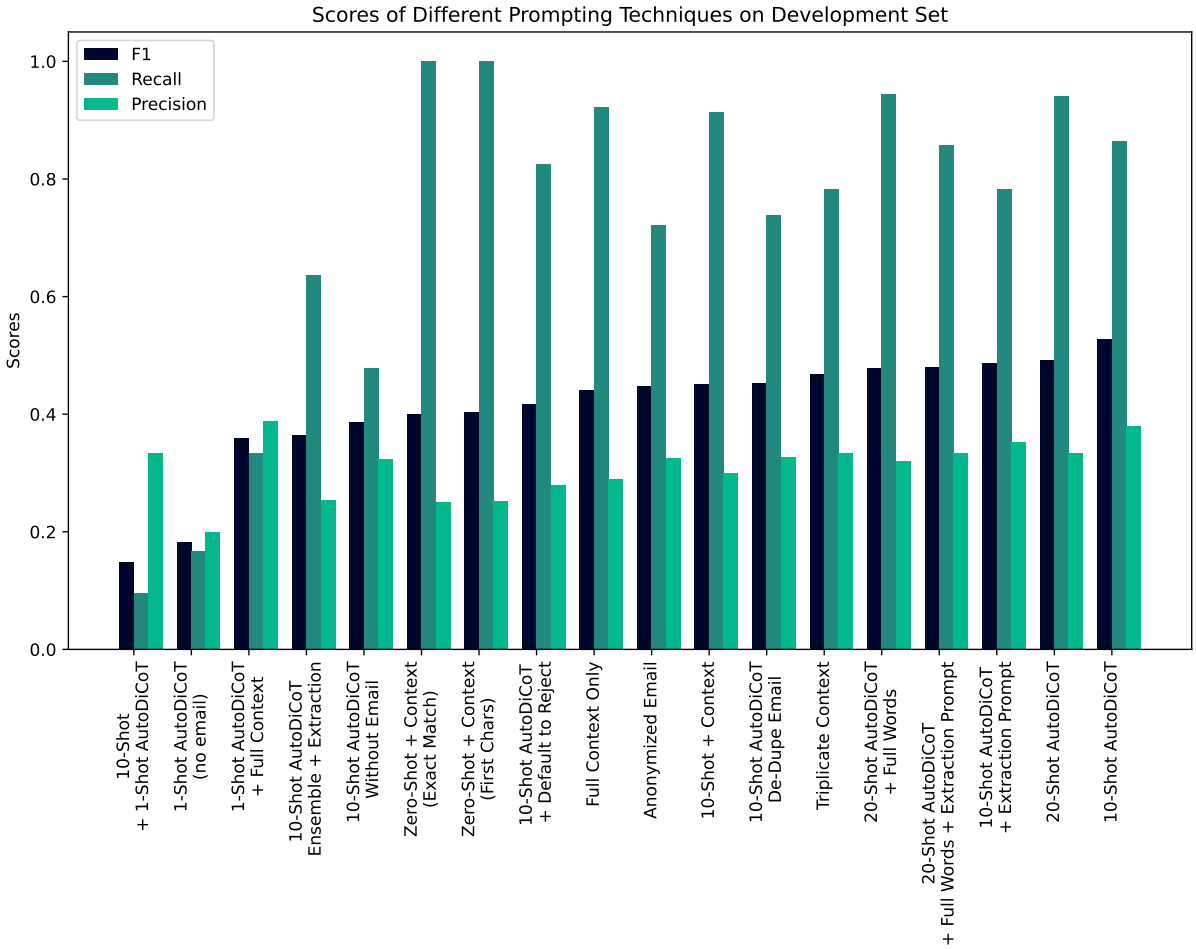

Figure 6.5: F1 scores varied widely from worst performing prompts to highest performing prompts, but most prompts scored within a similar range.

Chain-of-Thought, AutoCoT, Contrastive CoT, and multiple answer extraction techniques. We report statistics for the first runs of these techniques; F1 scores could change by as much as 0.04 upon subsequent runs, even with temperature and top p set to zero.[17]

Zero-Shot + Context was the first technique evaluated (Figure 6.10), using the description in Figure 6.7. Notice the word *definition* in the prompt, although Figure 6.7 is not a formal definition.

In order to obtain a final response from the LLM to use in calculating performance metrics, it was necessary to extract a label from the LLM output. The prompt engineer tested two extractors, one that checks if the output is exactly "Yes" or "No", and another which just checks if those words match the first few characters of the output. The latter had better performance, and it is used for the rest of this section until we reach CoT. This approach obtained 0.40 F1, 1.0 recall, and 0.25 precision, evaluated on all samples from the training/development since no samples had been used as exemplars.

10-Shot + Context. Next, the prompt engineer added the first ten data samples (with labels) into the prompt, in Q: (question) A: (answer) format (Figure 6.11). He evaluated this 10-shot prompt on the remaining items in the training/development set, yielding ↑0.05 (0.45) F1, ↓0.09 (0.91) recall, and ↑ 0.05 (0.30) precision, relative to the previous best prompt.[18]

One-Shot AutoDiCot + Full Context. After performing 10-shot prompting, the prompt engineer observed that the 12th item in the development set was being incorrectly being labeled as a positive instance, and began experimenting with ways of mod-

[17]Temperature and top-p are configuration hyperparameters that control randomness of the output (Schulhoff, 2022).

[18]Here and for the remainder of the case study, we judge "best" by F1, and we report on the current prompt under discussion relative to the best performing previous prompt.

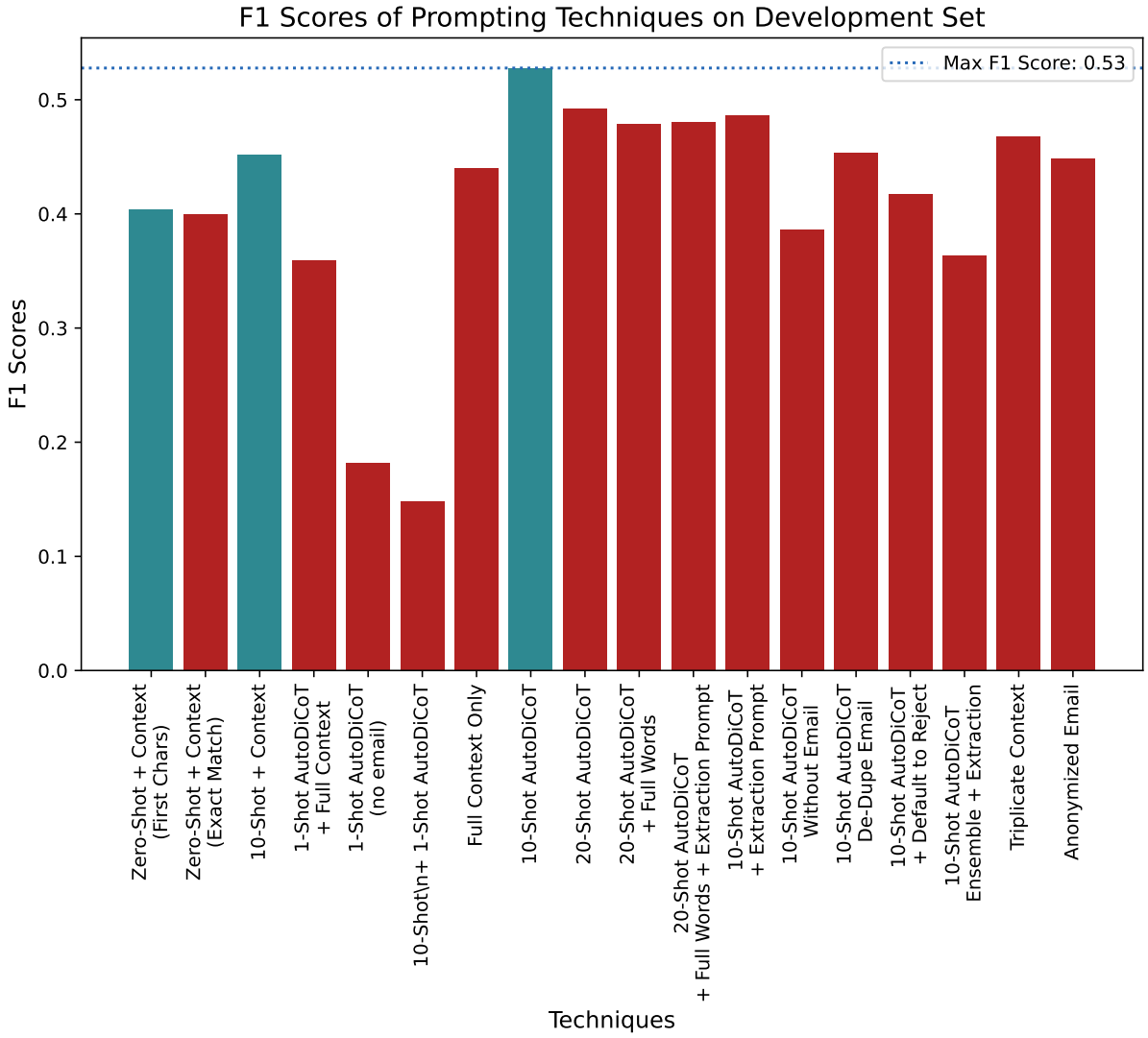

Figure 6.6: From the first prompt tried (Zero-Shot + Context) to the last (Anonymized Email), improvements in F1 score were hard to come by and and often involved testing multiple underperforming prompts before finding a performant one. Green lines show improvements over the current highest F1 score, while red lines show deteriorations.

Entrapment:
- Feeling like there is no exit
- Feeling hopeless
- Feeling like there is no way out
- Feeling afraid that things will never be normal again
- Feeling helpless to change
- Feeling trapped
- Feeling doomed
- Feeling or thinking that things will never change
- Feeling like there is no escape
- Feeling like there are no good solutions to problems

Figure 6.7: The description of entrapment used by the prompt engineer

What is entrapment with respect to Suicide Crisis Syndrome?

Figure 6.8: Question asked to the LLM to determine whether its training data had provided relevant knowledge about entrapment (it had not).

ifying the prompting such that the model would get that item correct. In order to get a sense of why this mislabeling was taking place, the prompt engineer prompted the LLM to generate an explanation of why the 12th item would have been labeled the way it was.[19]

[19]We are trying to avoid misleading language like "the LLM generated an explanation of its reasoning". LLMs do not have access to their own internal processes, and therefore they cannot "explain their reasoning" in the usual sense. An LLM generating an "explanation" is producing description of potential reasoning steps in getting to the output that could be true, but also may not be accurate at all.

If you're in immediate danger of harming yourself, please contact emergency services or a crisis hotline in your area. They can provide immediate support and help ensure your safety.

Figure 6.9: A snippet from an output, which does not label the data point, but rather attempts to provide mental health support to the user. Such outputs are often five times as long as this snippet.

{ENTRAPMENT DEFINITION (Figure 6.7)}
$\{q_{inf}\}$
Is this entrapment? Yes or no.

Figure 6.10: A Zero-Shot + Context prompt, the simplest of all prompts explored in this case study.

{ENTRAPMENT DEFINITION (Figure 6.7)}
Q: $\{q_1\}$
A: $\{a_1\}$
...
Q: $\{q_{10}\}$
A: $\{a_{10}\}$
Q: $\{q_{inf}\}$
A:

Figure 6.11: 10-Shot + Context Prompt

Figure 6.12 shows a version of that process, generalized to produce explanations for all development question/answer items (q_i, a_i) in a set T rather than just item 12. Informed by the reasoning steps r_{12} elicited with respect to the incorrectly labeled q_{12}, the previous prompt was modified by including r_{12} in a One-Shot CoT example with *incorrect* reasoning, as an exemplar for what *not* to do (Figure 6.13) .

We call the algorithm in Figure 6.12 Automatic Directed CoT (AutoDiCoT), since it automatically directs the CoT process to reason in a particular way. This technique can be generalized to any labeling task. It combines the automatic generation of CoTs (Zhang et al., 2022b) with showing the LLM examples of bad reasoning, as in the case of Contrastive CoT (Chia et al., 2023). The algorithm was also used in developing later prompts.

Finally, the prompt was extended with two additional pieces of context/instruction. The first was an email message the prompt engineer had received explaining overall goals of the project, which provided more context around the concept of entrapment and the reasons for wanting to label it. The second addition was inspired by the prompt engineer noticing the model was frequently overgenerating a positive label for entrapment. Hypothesizing that the model was being too aggressive in its pretraining-based inferences from the overt

1. Require: Development items T with n pairs (q_i, a_i)
2. For each pair (q_i, a_i) in T:
   (a) Label q_i as entrapment or not entrapment using the model
   (b) If the model labels correctly:
      i. Prompt the model with "Why?" to generate a reasoning chain r_i
   (c) Else:
      i. Prompt the model with "It is actually [is/is not] entrapment, please explain why." to generate a reasoning chain r_i
   (d) Store the tuple (q_i, r_i, a_i)
3. Return: n tuples (q_i, r_i, a_i)

Figure 6.12: Algorithm: Automatic Directed CoT

{PROFESSOR'S EMAIL}

{ENTRAPMENT DEFINITION (Figure 6.7)}

IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped.

Q: $\{q_{12}\}$
R: Although "Today I found out I have 10 days to vacate my apartment or I'll be formally evicted. I'm 2 months behind on my rent due to a bad time where I got demoted at work and rent from making roughly $1000 ever 2 weeks to around $450. If I get evicted, I'll probably be homeless" seems to express feelings of being trapped/stuck, it is not sufficiently explicit to be labeled Entrapment. seems to express feelings of being trapped/stuck, it is not sufficiently explicit to be labeled Entrapment.
A: $\{a_{12}\}$
Q: $\{q_{inf}\}$

Figure 6.13: One-Shot AutoDiCot + Full Context

language, he instructed the model to restrict itself to *explicit* statements of entrapment (Figure 6.13). Below we refer to these two pieces of context, provided in addition to the description of entrapment, as *full context*.

A new extractor was also used for this prompt, which checks if the last word in the output is "Yes" or "No", instead of the first word. This updated prompt was tested against all inputs in the development set except for the first 20. It did not improve F1, ↓0.09 (0.36) F1, but it led the prompt engineer in a direction that did, as discussed below. Recall dropped to ↓ 0.58 (0.33) recall and precision improved to ↑ 0.09 (0.39) precision.

At this point, though, it is worth observing that, although it did ultimately lead to a gain in F1 score, the steps taken here to cut down on over-generation of positive labels were not, in fact, the right move in terms of the longer term goals. Entrapment need *not* be expressed explicitly in order to be present (e.g. through phrases like "I feel trapped" or "There's no way out"); rather, clinical experts who have looked at the texts found that expressions of entrapment could be implicit and potentially quite nuanced. Moreover, in most use cases for automatically spotting entrapment in someone's language, precision and recall are unlikely to be equally important and, of the two, the recall/sensitivity (i.e. not missing people who should be flagged as at-risk) may matter more because the potential cost of a false negative is so high.

The take-away here, although the insight came later, is that it is easy for the process of prompt development to diverge from the actual goals unless regular engagement is fostered between the prompt engineer and domain experts who more deeply understand the real-world use case.

Ablating Email. The results of the previous changes were promising, but they did involve creating a prompt that included information from an email message that had not been created for that purpose, and which included information about the project, the dataset, etc. that were not intended for disclosure to a broad audience. Ironically, removing this email brought performance significantly down, ↓ 0.27 (0.18) F1, ↓ 0.75 (0.17) recall and ↓ 0.1 (0.20) precision. We attribute this to the fact that the email provided richer background information about the goals of the labeling. Although we

{PROFESSOR's EMAIL}

{ENTRAPMENT DEFINITION (Figure 6.7)}

IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped.

Q: $\{q_1\}$
A: $\{a_1\}$
...
Q: $\{q_{10}\}$
A: $\{a_{10}\}$
Q: $\{q_{12}\}$
R: Although "{LLM REASONING}" seems to express feelings of being trapped/stuck, it is not sufficiently explicit to be labeled Entrapment.
A: $\{a_{12}\}$
Q: $\{q_{inf}\}$

Figure 6.14: 10-Shot + 1 AutoDiCoT

would not recommend including email or any other potentially identifying information in any LLM prompt, we chose to leave the email in the prompt; this is consistent with scenarios in many typical settings, in which prompts are not expected to be exposed to others.

10-Shot + 1 AutoDiCoT. As a next step, the prompt engineer tried including full context, 10 regular exemplars, and the one-shot exemplar about how not to reason. This hurt performance (Figure 6.14) ↓ 0.30 (0.15) F1, ↓ 0.08 (0.10) recall, ↓ 0.03 (0.33) precision.

Full Context Only. Next, a prompt was created using only full context, without any exemplars (Figure 6.15). This boosted performance over the previous technique, but did not make progress overall ↓ 0.01 (0.44) F1, ↑ 0.01 (0.92) recall, ↓ 0.01 (0.29) precision. Interestingly, in this prompt, the prompt engineer accidentally pasted in the full-context email twice, and that ended up having significant positive effects on performance later (and removing the duplicate actually decreased performance). This is reminiscent of the re-reading technique (Xu et al., 2023).

{PROFESSOR's EMAIL}
{PROFESSOR's EMAIL}

{ENTRAPMENT DEFINITION (Figure 6.7)}

IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped.

Q: $\{q_{inf}\}$ A:

Figure 6.15: Full Context Only

This can be interpreted both optimistically and pessimistically. Optimistically, it demonstrates how improvements can arise through exploration and fortuitous discovery. On the pessimistic side, the value of duplicating the email in the prompt highlights the extent to which prompting remains a difficult to explain black art, where the LLM may turn out to be unexpectedly sensitive to variations one might not expect to matter.

10-Shot AutoDiCoT. The next step was to create more AutoDiCoT exemplars, per the algorithm in Figure 6.12. A total of ten new AutoDiCoT exemplars were added to the full context prompt (Figure 6.16). This yielded the most successful prompt from this prompt engineering exercise, in terms of F1 score, ↑ 0.08 (0.53) F1, ↓ 0.05 (0.86) recall, ↑ 0.08 (0.38) precision.

20-Shot AutoDiCoT. Further experimentation proceeded seeking (unsuccesfully) to improve on the previous F1 result. In one attempt, the prompt engineer labeled an additional ten exemplars, and created a 20-shot prompt from the first 20 data points in the development set. This led to worse results than the 10-shot prompt, when tested on all samples other than the first twenty, ↓ 0.04 (0.49) F1, ↑ 0.08 (0.94) recall, ↓ 0.05 (0.33) precision. Notably, it also yielded worse performance on the test set.

20-Shot AutoDiCoT + Full Words. The prompt engineer conjectured that the LLM would perform better if the prompt included full words *Question*, *Reasoning*, and *Answer* rather than *Q*, *R*, *A*. However, this did not succeed (Figure 6.17), ↓ 0.05

{PROFESSOR's EMAIL}

{ENTRAPMENT DEFINITION}

IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped.

Q: $\{q_1\}$
R: $\{r_1\}$
A: $\{a_1\}$
...
Q: $\{q_{10}\}$
R: $\{r_{10}\}$
A: $\{a_{10}\}$
Q: $\{q_{inf}\}$

Figure 6.16: 10-Shot AutoDiCoT

(0.48) F1, ↑ 0.08 (0.94) recall, ↓ 0.06 (0.32) precision.

20-Shot AutoDiCoT + Full Words + Extraction Prompt. The prompt engineer then noticed that in many cases, the LLM generated outputs that could not properly be parsed to obtain a response. So, they crafted a prompt that extracted answers from the LLM's response (Figure 6.18). Although this improved accuracy by a few points, it decreased F1, thanks to the fact that many of the outputs that had been unparsed actually contained incorrect responses, ↓ 0.05 (0.48) F1, ↓ 0.05 (0.33) precision, with no change in recall (0.86).

10-Shot AutoDiCoT + Extraction Prompt. Applying the extraction prompt to the best performing 10-Shot AutoDiCoT prompt did not improve results, ↓ 0.04 (0.49) F1, ↓ 0.08 (0.78) recall, ↓ 0.03 (0.35) precision.

10-Shot AutoDiCoT without Email. As noted above, removing the email outright from the prompt hurt performance, ↓ 0.14 (0.39) F1, ↓ 0.39 (0.48) recall, ↓ 0.06 (0.32) precision.

De-Duplicating Email. Also as noted above, it seemed reasonable that removing the duplication of the email would perform as well or better than the prompt with the unintentional duplication. As it turned out, however, removing the duplicate significantly hurt performance, ↓ 0.07 (0.45) F1, ↓ 0.12 (0.74) recall, ↓ 0.05 (0.33) precision.

{PROFESSOR's EMAIL}

{ENTRAPMENT DEFINITION}

IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped.

Question: $\{q_1\}$
Reasoning: $\{r_1\}$
Answer: $\{a_1\}$
...
Question: $\{q_{20}\}$
Reasoning: $\{r_{20}\}$
Answer: $\{a_{20}\}$
Question: $\{q_{inf}\}$

Figure 6.17: 20-shot AutoDiCoT

{PROFESSOR's EMAIL}

{ENTRAPMENT DEFINITION}

IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped.

Question: {REDACTED}
Answer: {ANSWER}

Does this Answer indicate entrapment? Output the word Yes if it is labeled as entrapment and output the word No if it is not labeled as entrapment. Only output the word Yes or the word No.

Figure 6.18: Extraction Prompt

10-Shot AutoDiCoT + Default to Reject. This approach used the best performing prompt, and defaulted to labeling as negative (not entrapment) in the case of answers that are not extracted properly. This did not help performance, ↓ 0.11 (0.42) F1, ↓ 0.04 (0.83) recall, ↓ 0.10 (0.28) precision.

Ensemble + Extraction. Especially for systems that are sensitive to the details of their inputs, there are advantages in trying multiple variations of an input and then combining their results. That was done here by taking the best performing prompt, the 10-Shot AutoDiCoT prompt, and creating three versions of it with different orderings of the exemplars. The average of the three results was taken to be the final answer. Unfortunately, both orderings that differed from the default ordering led to the LLM not outputting a well-structured response. An extraction prompt was therefore used to obtain final answers. This exploration hurt rather than helped performance ↓ 0.16 (0.36) F1, ↓ 0.23 (0.64) recall, ↓ 0.13 (0.26) precision.

10-Shot AutoCoT + 3x the context (no email dupe). Recall that *context* refers to the description of entrapment, an instruction about explicitness, and an email. Since the duplicated email had improved performance, the prompt engineer tested out pasting in three copies of the context (first de-duplicating the email). However, this did not improve performance, ↓ 0.06 (0.47) F1, ↓ 0.08 (0.78) recall, ↓ 0.05 (0.33) precision.

Anonymize Email. At this point it seemed clear that including the duplicated email in the prompt was actually, although not explainably, essential to the best performance so far obtained. The prompt engineer decided to anonymize the email by replacing personal names with other, random names. However, surprisingly, this decreased performance significantly ↓ 0.08 (0.45) F1, ↓ 0.14 (0.72) recall, ↓ 0.06 (0.33) precision.

DSPy. We concluded the case study by exploring an alternative to manual prompt engineering, the DSPy framework (Khattab et al., 2023), which automatically optimizes LLM prompts for a given target metric. Specifically, we begin with a chain-of-thought classification pipeline that uses the definition of entrapment in Figure 6.7. Over 16 iterations, DSPy bootstrapped synthetic LLM-generated demonstrations and randomly sampled training exemplars, with the ultimate objective of maximizing $F1$ on the same development set used above. We used `gpt-4-0125-preview` and the default settings for the `BootstrapFewShotWithRandomSearch` "teleprompter" (the optimization approach). Figure 6.19 shows the results of two of these prompts on the test set, one of which used default DSPy behaviour, and the second which was manually modified slightly from this default. The best resulting prompt includes 15 exemplars (without CoT reasoning) and one bootstrapped reasoning demonstration. It achieves 0.548 $F1$ (and 0.385 / 0.952 precision / recall) on the test set, without making any use of the professor's email nor the incorrect instruction about the explicitness of entrapment. It also performs much better than the human prompt engineer's prompts on the test set, which demonstrates the significant promise of automated prompt engineering.

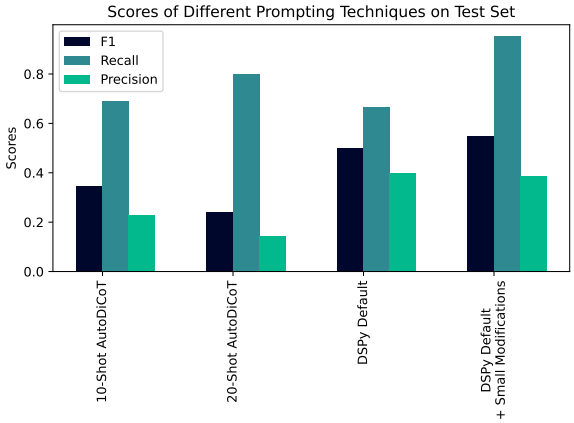

Figure 6.19: Scores of different prompting techniques on the test set.

6.2.4 Discussion

Prompt engineering is a non-trivial process, the nuances of which are not currently well described in literature. From the fully manual process illustrated above, there are several take-aways worth summarizing. First, prompt engineering is fundamentally different from other ways of getting a computer to behave the way you want it to: these systems are being cajoled, not programmed, and, in addition to being quite sensitive to the specific LLM being used, they can be incredibly sensitive to specific details in prompts without there being any obvious reason those details should matter. Second, therefore, it is important to dig into the data (e.g. generating potential explanations for LLM "reasoning" that leads to incorrect responses). Related, the

third and most important take-away is that prompt engineering should involve engagement between the prompt engineer, who has expertise in how to coax LLMs to behave in desired ways, and domain experts, who understand what those desired ways are and why.

Ultimately we found that there was significant promise in an automated method for exploring the prompting space, but also that combining that automation with human prompt engineering/revision was the most successful approach. We hope that this study will serve as a step toward more robust examinations of how to perform prompt engineering.

7 Related Work

In this section, we review existing surveys and meta-analyses of prompting. Liu et al. (2023b) perform a systematic review of prompt engineering in the pre-ChatGPT era, including various aspects of prompting like prompt template engineering, answer engineering, prompt ensembling, and prompt tuning methods. Their review covers many different types of prompting (e.g., cloze, soft-prompting, etc., across many different types of language models) while we focus on discrete pre-fix prompting but more in-depth discussion. Chen et al. (2023a) provide a review of popular prompting techniques like Chain-of-Thought, Tree-of-Thought, Self-Consistency, and Least-to-Most prompting, along with outlooks for future prompting research. White et al. (2023) and Schmidt et al. (2023) provide a taxonomy of prompt patterns, which are similar to software patterns (and prompting techniques for that matter). Gao (2023) provide a practical prompting technique tutorial for a non-technical audience. Santu and Feng (2023) provide a general taxonomy of prompts that can be used to design prompts with specific properties to perform a wide range of complex tasks. Bubeck et al. (2023) qualitatively experiment with a wide range of prompting methods on the early version of GPT-4 to understand its capabilities. Chu et al. (2023) review Chain-of-Thought related prompting methods for reasoning. In earlier work, Bommasani et al. (2021) review and discuss opportunities and risks of foundation models broadly, and Dang et al. (2022) discuss prompting strategies for interactive creative applications that use prompting as a new paradigm for human interaction, with a particular focus on the user interface design that supports user prompting. As an addition to these existing surveys, our review aims to provide a more updated and formalized systematic review.

There is also a line of work that surveys prompting techniques for particular domains or downstream applications. Meskó (2023) and Wang et al. (2023d) offer recommended use cases and limitations of prompt engineering in the medical and healthcare domains. Heston and Khun (2023) provide a review of prompt engineering for medical education use cases. Peskoff and Stewart (2023) query ChatGPT and YouChat to assess domain coverage. Hua et al. (2024) use a GPT-4-automated approach to review LLMs in the mental health space. Wang et al. (2023c) review prompt engineering and relevant models in the visual modality and Yang et al. (2023e) provided a comprehensive list of qualitative analyses of multimodal prompting, particularly focusing on GPT-4V[20]. Durante et al. (2024) review multimodal interactions based on LLM embodied agents. Ko et al. (2023b) review literature on the adoption of Text-to-Image generation models for visual artists' creative works. Gupta et al. (2024) review GenAI through a topic modeling approach. Awais et al. (2023) review foundation models in vision, including various prompting techniques. Hou et al. (2023) perform a systematic review of prompt engineering techniques as they relate to software engineering. They use a systematic review technique developed by Keele et al. (2007), specifically for software engineering reviews. Wang et al. (2023e) review the literature on software testing with large language models. Zhang et al. (2023a) review ChatGPT prompting performance on software engineering tasks such as automated program repair. Neagu (2023) provide a systematic review on how prompt engineering can be leveraged in computer science education. Li et al. (2023j) review literature on the fairness of large language models. There are also surveys on related aspects such as hallucination of language models (Huang et al., 2023b), verifiability (Liu et al., 2023a), reasoning (Qiao et al., 2022), augmentation (Mialon et al., 2023), and linguistic properties of prompts (Leidinger et al., 2023). Different from these works, we perform our review targeting broad coverage and generally applicable prompting techniques. Finally, in terms of more general prior and concurrent surveys (Liu et al., 2023b; Sahoo et al., 2024; Vatsal and Dubey, 2024), this survey offers an update in a fast-moving field. In addition, we provide a starting point for taxonomic organization of prompting techniques and standardization of terminology. Moreover, unlike many works that claim to be systematic, we base our work in the widely used standard for systematic literature reviews—PRISMA (Page et al., 2021).

[20] https://openai.com/research/gpt-4v-system-card

8 Conclusions

Generative AI is a novel technology, and broader understanding of models' capabilities and limitations remains limited. Natural language is a flexible, open-ended interface, with models having few obvious affordances. The use of Generative AI therefore inherits many of the standard challenges of linguistic communication—e.g., ambiguity, the role of context, the need for course correction—while at the same time adding the challenge of communicating with an entity whose "understanding" of language may not bear any substantial relationship to human understanding. Many of the techniques described here have been called "emergent", but it is perhaps more appropriate to say that they were *discovered*—the result of thorough experimentation, analogies from human reasoning, or pure serendipity.

The present work is an initial attempt to categorize the species of an unfamiliar territory. While we make every attempt to be comprehensive, there are sure to be gaps and redundancies. Our intention is to provide a taxonomy and terminology that cover a large number of existing prompt engineering techniques, and which can accommodate future methods. We discuss over 200 prompting techniques, frameworks built around them, and issues like safety and security that need to be kept in mind when using them. We also present two case studies in order to provide a clear sense of models' capabilities and what it is like to tackle a problem in practice. Last, our stance is primarily observational, and we make no claims to the validity of the presented techniques. The field is new, and evaluation is variable and unstandardized—even the most meticulous experimentation may suffer from unanticipated shortcomings, and model outputs themselves are sensitive to meaning-preserving changes in inputs. As a result, we encourage the reader to avoid taking any claims at face value and to recognize that techniques may not transfer to other models, problems, or datasets.

To those just beginning in prompt engineering, our recommendations resemble what one would recommend in any machine learning setting: understand the *problem* you are trying to solve (rather than just focusing on input/output and benchmark scores), and ensure the data and metrics you are working with constitute a good representation of that problem. It is better to start with simpler approaches first, and to remain skeptical of claims about method performance. To those already engaged in prompt engineering, we hope that our taxonomy will shed light on the relationships between existing techniques. To those developing new techniques, we encourage situating new methods within our taxonomy, as well as including ecologically valid case studies and illustrations of those techniques.

Acknowledgements

We appreciate the advice given by Hal Daumé III, Adam Visokay, and Jordan Boyd-Graber and review by Diyi Yang, Brandon M. Stewart, Shubham Vatsal, Mason Marchetti, Aaron Tay, Andrea Vella, and Allie Miller. We also appreciate the 10K USD in API credits given by OpenAI and design work by Benjamin DiMarco.

References

Adept. 2023. ACT-1: Transformer for Actions. https://www.adept.ai/blog/act-1.

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. 2024. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*.

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8857–8873, Toronto, Canada. Association for Computational Linguistics.

Kabir Ahuja, Harshita Diddee, Rishav Hada, Millicent Ochieng, Krithika Ramesh, Prachi Jain, Akshay Nambi, Tanuja Ganu, Sameer Segal, Maxamed Axmed, Kalika Bali, and Sunayana Sitaram. 2023. MEGA: Multilingual Evaluation of Generative AI. In *EMNLP*.

Rebuff AI. 2023. A self-hardening prompt injection detector.

Anirudh Ajith, Chris Pan, Mengzhou Xia, Ameet Deshpande, and Karthik Narasimhan. 2024. InstructEval: Systematic evaluation of instruction selection methods. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4336–4350, Mexico City, Mexico. Association for Computational Linguistics.

Sílvia Araújo and Micaela Aguiar. 2023. Comparing chatgpt's and human evaluation of scientific texts' translations from english to portuguese using popular automated translators. *CLEF*.

ArthurAI. 2024. Arthur shield.

Akari Asai, Sneha Kudugunta, Xinyan Velocity Yu, Terra Blevins, Hila Gonen, Machel Reid, Yulia Tsvetkov, Sebastian Ruder, and Hannaneh Hajishirzi. 2023. BUFFET: Benchmarking Large Language Models for Few-shot Cross-lingual Transfer.

Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. 2023. Foundational models defining a new era in vision: A survey and outlook.

Abhijeet Awasthi, Nitish Gupta, Bidisha Samanta, Shachi Dave, Sunita Sarawagi, and Partha Talukdar. 2023. Bootstrapping multilingual semantic parsers using large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2455–2467, Dubrovnik, Croatia. Association for Computational Linguistics.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023a. Longbench: A bilingual, multitask benchmark for long context understanding.

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2023b. Benchmarking Foundation Models with Language-Model-as-an-Examiner. In *NeurIPS 2023 Datasets and Benchmarks*.

Chris Bakke. 2023. Buying a chevrolet for 1$.

Nishant Balepur, Jie Huang, and Kevin Chang. 2023. Expository text generation: Imitate, retrieve, paraphrase. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11896–11919, Singapore. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. In *AACL*.

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. 2023. Rethinking the Role of Scale for In-Context Learning: An Interpretability-based Case Study at 66 Billion Scale. In *ACL*.

Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. 2022. Text2live: Text-driven layered image and video editing.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. 2024. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna,

Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel J. Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the Opportunities and Risks of Foundation Models. *ArXiv*, abs/2108.07258.

Hezekiah J. Branch, Jonathan Rodriguez Cefalu, Jeremy McHugh, Leyla Hujer, Aditya Bahl, Daniel del Castillo Iglesias, Ron Heichman, and Ramesh Darwishi. 2022. Evaluating the susceptibility of pre-trained language models via handcrafted adversarial examples.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym.

Tim Brooks, Bill Peebles, Connor Homes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Wing Yin Ng, Ricky Wang, and Aditya Ramesh. 2024. Video generation models as world simulators. *OpenAI*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, John A. Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuan-Fang Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models.

CDC. 2023. Suicide data and statistics.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Ernie Chang, Pin-Jie Lin, Yang Li, Sidd Srinivasan, Gael Le Lan, David Kant, Yangyang Shi, Forrest Iandola, and Vikas Chandra. 2023. In-context prompt editing for conditional audio generation.

Harrison Chase. 2022. LangChain.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023a. Unleashing the potential of prompt engineering in large language models: a comprehensive review.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. How is chatgpt's behavior changing over time? *arXiv preprint arXiv:2307.09009*.

Shiqi Chen, Siyang Gao, and Junxian He. 2023c. Evaluating factual consistency of summaries with large language models. *arXiv preprint arXiv:2305.14069*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023d. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *TMLR*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023e. Universal self-consistency for large language model generation.

Yang Chen, Yingwei Pan, Yehao Li, Ting Yao, and Tao Mei. 2023f. Control3d: Towards controllable text-to-3d generation.

Yi Chen, Rui Wang, Haiyun Jiang, Shuming Shi, and Ruifeng Xu. 2023g. Exploring the use of large language models for reference-free text quality evaluation: An empirical study. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pages 361–374, Nusa Dua, Bali. Association for Computational Linguistics.

Jiaxin Cheng, Tianjun Xiao, and Tong He. 2023. Consistent video-to-video transfer using synthetic dataset. *ArXiv*, abs/2311.00213.

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. 2023. Contrastive chain-of-thought prompting.

Jiqun Chu and Zuoquan Lin. 2023. Entangled representation learning: A bidirectional encoder decoder model. In *Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI '22, New York, NY, USA. Association for Computing Machinery.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future.

Robert J Cramer, Jacinta Hawgood, Andréa R Kaniuka, Byron Brooks, and Justin C Baker. 2023. Updated suicide prevention core competencies for mental health professionals: Implications for training, research, and practice. *Clinical Psychology: Science and Practice*.

Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. 2022. Vqgan-clip: Open domain image generation and editing with natural language guidance.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Hai Dang, Lukas Mecke, Florian Lehmann, Sven Goller, and Daniel Buschek. 2022. How to prompt? opportunities and challenges of zero- and few-shot learning for human-ai interaction in creative applications of generative models.

Maksym Del and Mark Fishel. 2023. True detective: A deep abductive reasoning benchmark undoable for gpt-3 and challenging for gpt-4. In *Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*. Association for Computational Linguistics.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning. In *RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning*.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models.

Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models.

Ming Ding, Zhuoyi Yang, Wenyi Hong, Wendi Zheng, Chang Zhou, Da Yin, Junyang Lin, Xu Zou, Zhou Shao, Hongxia Yang, and Jie Tang. 2021. Cogview: Mastering text-to-image generation via transformers. In *Advances in Neural Information Processing Systems*, volume 34, pages 19822–19835. Curran Associates, Inc.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, Jie Meng, Wenjie Ruan, and Xiaowei Huang. 2024. Building guardrails for large language models.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. In *NeurIPS*.

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, Katsushi Ikeuchi, Hoi Vo, Fei-Fei Li, and Jianfeng Gao. 2024. Agent ai: Surveying the horizons of multimodal interaction.

Julen Etxaniz, Gorka Azkune, Aitor Soroa, Oier Lopez de Lacalle, and Mikel Artetxe. 2023. Do multilingual language models think better in english?

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:594–611.

Lincong Feng, Muyu Wang, Maoyu Wang, Kuo Xu, and Xiaoli Liu. 2023. Metadreamer: Efficient text-to-3d creation with disentangling geometry and texture.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. 2023. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1066–1083, Singapore. Association for Computational Linguistics.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023a. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Jinlan Fu, See-Kiong Ng, and Pengfei Liu. 2022. Polyglot prompt: Multilingual multitask prompt training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9919–9935, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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*.

Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. 2011. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas Liao, Kamilė Lukošiūtė, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, et al. 2023. The capacity for moral self-correction in large language models. *arXiv preprint arXiv:2302.07459*.

Andrew Gao. 2023. Prompt engineering for large language models. *SSRN*.

Lingyu Gao, Aditi Chaudhary, Krishna Srinivasan, Kazuma Hashimoto, Karthik Raman, and Michael Bendersky. 2023a. Ambiguity-aware in-context learning with large language models. *arXiv preprint arXiv:2309.07900*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Mingqi Gao, Jie Ruan, Renliang Sun, Xunjian Yin, Shiping Yang, and Xiaojun Wan. 2023c. Human-like summarization evaluation with chatgpt. *arXiv preprint arXiv:2304.02554*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Marisa Garcia. 2024. What air canada lost in 'remarkable' lying ai chatbot case. *Forbes*.

Xavier Garcia, Yamini Bansal, Colin Cherry, George Foster, Maxim Krikun, Melvin Johnson, and Orhan Firat. 2023. The unreasonable effectiveness of few-shot learning for machine translation. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

MF Garnett and SC Curtin. 2023. Suicide mortality in the united states, 2001–2021. *NCHS Data Brief*, 464:1–8.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92.

Marjan Ghazvininejad, Hila Gonen, and Luke Zettlemoyer. 2023. Dictionary-based phrase-level prompting of large language models for machine translation.

Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. 2023. Emu video: Factorizing text-to-video generation by explicit image conditioning.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. 2023. Figstep: Jailbreaking large vision-language models via typographic visual prompts.

Riley Goodside. 2022. Exploiting gpt-3 prompts with malicious inputs that order the model to ignore its previous directions.

Google. 2023. Gemini: A family of highly capable multimodal models.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024a. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024b. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Han Guo, Bowen Tan, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. 2022. Efficient (soft) q-learning for text generation with limited good data.

Priyanka Gupta, Bosheng Ding, Chong Guan, and Ding Ding. 2024. Generative ai: A systematic review using topic modelling techniques. *Data and Information Management*, page 100066.

Rishav Hada, Varun Gumma, Adrian Wynter, Harshita Diddee, Mohamed Ahmed, Monojit Choudhury, Kalika Bali, and Sunayana Sitaram. 2024. Are large language model-based evaluators the solution to scaling up multilingual evaluation? In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1051–1070, St. Julian's, Malta. Association for Computational Linguistics.

Muhammad Usman Hadi, Qasem Al Tashi, Rizwan Qureshi, Abbas Shah, Amgad Muneer, Muhammad Irfan, and et al. 2023. Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. *TechRxiv*.

Aparna Dhinakaran Hakan Tekgul. 2023. Guardrails: What are they and how can you use nemo and guardrails ai to safeguard llms? Online.

Sherzod Hakimov and David Schlangen. 2023. Images in language space: Exploring the suitability of large language models for vision & language tasks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14196–14210, Toronto, Canada. Association for Computational Linguistics.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting Frozen Language Models with Massive Tools via Tool Embeddings. In *NeurIPS*.

Hangfeng He, Hongming Zhang, and Dan Roth. 2023a. Socreval: Large language models with the socratic method for reference-free reasoning evaluation. *arXiv preprint arXiv:2310.00074*.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2023b. Exploring human-like translation strategy with large language models.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *ICLR*.

Amr Hendy, Mohamed Gomaa Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *ArXiv*, abs/2302.09210.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2022. Prompt-to-prompt image editing with cross attention control.

T.F. Heston and C. Khun. 2023. Prompt engineering in medical education. *Int. Med. Educ.*, 2:198–205.

Tobias Hinz, Stefan Heinrich, and Stefan Wermter. 2022. Semantic object accuracy for generative text-to-image synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1552–1565.

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2023. Large language models for software engineering: A systematic literature review.

Ming-Hao Hsu, Kai-Wei Chang, Shang-Wen Li, and Hung yi Lee. 2023. An exploration of in-context learning for speech language model.

Yining Hua, Fenglin Liu, Kailai Yang, Zehan Li, Yi han Sheu, Peilin Zhou, Lauren V. Moran, Sophia Ananiadou, and Andrew Beam. 2024. Large language models in mental health care: a scoping review.

Haoyang Huang, Tianyi Tang, Dongdong Zhang, Wayne Xin Zhao, Ting Song, Yan Xia, and Furu Wei. 2023a. Not all languages are created equal in llms: Improving multilingual capability by cross-lingual-thought prompting.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023b. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions.

Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, Qiang Liu, Kriti Aggarwal, Zewen Chi, Johan Bjorck, Vishrav Chaudhary, Subhojit Som, Xia Song, and Furu Wei. 2023c. Language is not all you need: Aligning perception with language models.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations.

Vivek Iyer, Pinzhen Chen, and Alexandra Birch. 2023. Towards effective disambiguation for machine translation with large language models.

Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. 2022. Zero-shot text-guided object generation with dream fields.

Qi Jia, Siyu Ren, Yizhu Liu, and Kenny Q Zhu. 2023. Zero-shot faithfulness evaluation for text summarization with foundation language model. *arXiv preprint arXiv:2310.11648*.

Yixing Jiang, Jeremy Irvin, Ji Hun Wang, Muhammad Ahmed Chaudhry, Jonathan H Chen, and Andrew Y Ng. 2024. Many-shot in-context learning in multimodal foundation models. *arXiv preprint arXiv:2405.09798*.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine.

Ziqi Jin and Wei Lu. 2023. Tab-cot: Zero-shot tabular chain of thought.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, Dor Muhlgay, Noam Rozen, Erez Schwartz, Gal Shachaf, Shai Shalev-Shwartz, Amnon Shashua, and Moshe Tenenholtz. 2022. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning.

Staffs Keele et al. 2007. Guidelines for performing systematic literature reviews in software engineering.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Kimiya Keyvan and Jimmy Xiangji Huang. 2022. How to approach ambiguous queries in conversational search: A survey of techniques, approaches, tools, and challenges. *ACM Computing Surveys*, 55(6):1–40.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Exploring demonstration ensembling for in-context learning.

Mahmoud Khalil, Ahmad Khalil, and Alioune Ngom. 2023. A comprehensive study of vision transformers in image classification tasks.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks.

Natalie Kiesler and Daniel Schiffner. 2023. Large language models in introductory programming education: Chatgpt's performance and implications for assessments. *arXiv preprint arXiv:2308.08572*.

Hwichan Kim and Mamoru Komachi. 2023. Enhancing few-shot cross-lingual transfer with target language peculiar examples. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 747–767, Toronto, Canada. Association for Computational Linguistics.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang goo Lee. 2022. Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator.

Sunkyoung Kim, Dayeon Ki, Yireun Kim, and Jinsik Lee. 2023. Boosting cross-lingual transferability in multilingual models via in-context learning.

Dayoon Ko, Sangho Lee, and Gunhee Kim. 2023a. Can language models laugh at youtube short-form videos?

Hyung-Kwon Ko, Gwanmo Park, Hyeon Jeon, Jaemin Jo, Juho Kim, and Jinwook Seo. 2023b. Large-scale text-to-image generation models for visual artists' creative works. *Proceedings of the 28th International Conference on Intelligent User Interfaces*.

Tom Kocmi and Christian Federmann. 2023a. Gemba-mqm: Detecting translation quality error spans with gpt-4. *arXiv preprint arXiv:2310.13988*.

Tom Kocmi and Christian Federmann. 2023b. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, Tampere, Finland. European Association for Machine Translation.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners.

Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot learning.

Will Kurt. 2024. Say what you mean: A response to 'let me speak freely'. `https://blog.dottxt.co/say-what-you-mean.html`.

Gihyun Kwon and Jong Chul Ye. 2022. Clipstyler: Image style transfer with a single text condition.

Lakera. 2024. Lakera guard.

Bar Lanyado, Ortal Keizman, and Yair Divinsky. 2023. Can you trust chatgpt's package recommendations? Vulcan Cyber Blog.

Cindy Le, Congrui Hetang, Ang Cao, and Yihui He. 2023. Euclidreamer: Fast and high-quality texturing for 3d models with stable diffusion depth.

Soochan Lee and Gunhee Kim. 2023. Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models.

Alina Leidinger, Robert van Rooij, and Ekaterina Shutova. 2023. The language of prompting: What linguistic properties make a prompt successful?

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. 2019a. Controllable text-to-image generation.

Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. 2023a. Large language models understand and can be enhanced by emotional stimuli.

Chengzhengxu Li, Xiaoming Liu, Yichen Wang, Duyi Li, Yu Lan, and Chao Shen. 2023b. Dialogue for prompting: a policy-gradient-based discrete prompt optimization for few-shot learning.

Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. 2023c. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model.

Ming Li, Pan Zhou, Jia-Wei Liu, Jussi Keppo, Min Lin, Shuicheng Yan, and Xiangyu Xu. 2023d. Instant3d: Instant text-to-3d generation.

Ruosen Li, Teerth Patel, and Xinya Du. 2023e. Prd: Peer rank and discussion improve large language model based evaluations. *arXiv preprint arXiv:2307.02762*.

Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. 2019b. Object-driven text-to-image synthesis via adversarial training.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023f. Unified demonstration retriever for in-context learning.

Xiaonan Li and Xipeng Qiu. 2023a. Finding support examples for in-context learning.

Xiaonan Li and Xipeng Qiu. 2023b. Mot: Memory-of-thought enables chatgpt to self-improve.

Xiaoqian Li, Ercong Nie, and Sheng Liang. 2023g. Crosslingual retrieval augmented in-context learning for bangla.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks.

Yaoyiran Li, Anna Korhonen, and Ivan Vulić. 2023h. On bilingual lexicon induction with large language models.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023i. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Yingji Li, Mengnan Du, Rui Song, Xin Wang, and Ying Wang. 2023j. A survey on fairness in large language models.

Jingyun Liang, Yuchen Fan, Kai Zhang, Radu Timofte, Luc Van Gool, and Rakesh Ranjan. 2023. Movideo: Motion-aware video generation with diffusion models.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2023. Magic3d: High-resolution text-to-3d content creation.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona Diab, Veselin Stoyanov, and Xian Li. 2022. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yen-Ting Lin and Yun-Nung Chen. 2023. Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. *arXiv preprint arXiv:2305.13711*.

Jerry Liu. 2022. LlamaIndex.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for GPT-3? In *Workshop on Knowledge Extraction and Integration for Deep Learning Architectures; Deep Learning Inside Out*.

Nelson F Liu, Tianyi Zhang, and Percy Liang. 2023a. Evaluating verifiability in generative search engines. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

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

Weihuang Liu, Xi Shen, Chi-Man Pun, and Xiaodong Cun. 2023c. Explicit visual prompting for low-level structure segmentations. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023d. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Yihao Liu, Xiangyu Chen, Xianzheng Ma, Xintao Wang, Jiantao Zhou, Yu Qiao, and Chao Dong. 2023e. Unifying image processing as visual prompting question answering.

Yongkang Liu, Shi Feng, Daling Wang, Yifei Zhang, and Hinrich Schütze. 2023f. Evaluate what you can't evaluate: Unassessable generated responses quality. *arXiv preprint arXiv:2305.14658*.

Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. 2023g. Text-guided texturing by synchronized multi-view diffusion.

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2023h. Calibrating llm-based evaluator. *arXiv preprint arXiv:2309.13308*.

Jieyi Long. 2023. Large language model guided tree-of-thought.

Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. 2023. Att3d: Amortized text-to-3d object synthesis.

Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2023a. Bounding the capabilities of large language models in open text generation with prompt constraints.

Hongyuan Lu, Haoyang Huang, Dongdong Zhang, Haoran Yang, Wai Lam, and Furu Wei. 2023b. Chain-of-dictionary prompting elicits translation in large language models.

Qingyu Lu, Baopu Qiu, Liang Ding, Liping Xie, and Dacheng Tao. 2023c. Error analysis prompting enables human-like translation evaluation in large language models: A case study on chatgpt. *arXiv preprint arXiv:2303.13809*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity.

Yao Lu, Jiayi Wang, Raphael Tang, Sebastian Riedel, and Pontus Stenetorp. 2024. Strings from the library of babel: Random sampling as a strong baseline for prompt optimisation.

Charles Duffy Luca Beurer-Kellner, Marc Fischer. 2023. lmql. GitHub repository.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for abstractive text summarization. *arXiv preprint arXiv:2303.15621*.

Jiaxi Lv, Yi Huang, Mingfu Yan, Jiancheng Huang, Jianzhuang Liu, Yifan Liu, Yafei Wen, Xiaoxin Chen, and Shifeng Chen. 2023. Gpt4motion: Scripting physical motions in text-to-video generation via blender-oriented gpt planning.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning.

Huan Ma, Changqing Zhang, Yatao Bian, Lemao Liu, Zhirui Zhang, Peilin Zhao, Shu Zhang, Huazhu Fu, Qinghua Hu, and Bingzhe Wu. 2023. Fairness-guided few-shot prompting for large language models. *arXiv preprint arXiv:2303.13217*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35.

Laura Melzer, Thomas Forkmann, and Tobias Teismann. 2024. Suicide crisis syndrome: A systematic review. *Suicide and Life-Threatening Behavior*. February 27, online ahead of print.

Fanxu Meng, Haotong Yang, Yiding Wang, and Muhan Zhang. 2023. Chain of images for intuitively reasoning.

B. Meskó. 2023. Prompt engineering as an important emerging skill for medical professionals: Tutorial. *Journal of Medical Internet Research*, 25(Suppl 1):e50638.

Yachun Mi, Yu Li, Yan Shu, Chen Hui, Puchao Zhou, and Shaohui Liu. 2023. Clif-vqa: Enhancing video quality assessment by incorporating high-level semantic information related to human feelings.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work?

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

R.A. Morelli, J.D. Bronzino, and J.W. Goethe. 1991. A computational speech-act model of human-computer conversations. In *Proceedings of the 1991 IEEE Seventeenth Annual Northeast Bioengineering Conference*, pages 263–264.

Yasmin Moslem, Rejwanul Haque, John D. Kelleher, and Andy Way. 2023. Adaptive machine translation with large language models. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 227–237, Tampere, Finland. European Association for Machine Translation.

Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, Chenxue Wang, Shichao Liu, and Qing Wang. 2023. Clarifygpt: Empowering llm-based code generation with intention clarification.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15991–16111, Toronto, Canada. Association for Computational Linguistics.

Akshay Nambi, Vaibhav Balloli, Mercy Ranjit, Tanuja Ganu, Kabir Ahuja, Sunayana Sitaram, and Kalika Bali. 2023. Breaking language barriers with a leap: Learning strategies for polyglot llms.

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable extraction of training data from (production) language models.

National Center for Health Workforce Analysis. 2023. Behavioral health workforce, 2023.

Alexandra Neagu. 2023. How can large language models and prompt engineering be leveraged in Computer Science education?: Systematic literature review. Master's thesis, Delft University of Technology, 6.

Ercong Nie, Sheng Liang, Helmut Schmid, and Hinrich Schütze. 2023. Cross-lingual retrieval augmented prompt for low-resource languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8320–8340, Toronto, Canada. Association for Computational Linguistics.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding.

OpenAI. 2023. OpenAI Assistants.

Jonas Oppenlaender. 2023. A taxonomy of prompt modifiers for text-to-image generation.

Anton Osika. 2023. gpt-engineer.

Matthew J Page, Joanne E McKenzie, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, Roger Chou, Julie Glanville, Jeremy M Grimshaw, Asbjørn Hróbjartsson, Manoj M Lalu, Tianjing Li, Elizabeth W Loder, Evan Mayo-Wilson, Steve McDonald, Luke A McGuinness, Lesley A Stewart, James Thomas, Andrea C Tricco, Vivian A Welch, Penny Whiting, and David Moher. 2021. The prisma 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, 372.

Ehsan Pajouheshgar, Yitao Xu, Alexander Mordvintsev, Eyvind Niklasson, Tong Zhang, and Sabine Süsstrunk. 2023. Mesh neural cellular automata.

Pruthvi Patel, Swaroop Mishra, Mihir Parmar, and Chitta Baral. 2022. Is a question decomposition unit all we need?

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *ArXiv*, abs/2305.15334.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2021. Asleep at the keyboard? assessing the security of github copilot's code contributions.

Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. 2022. Examining zero-shot vulnerability repair with large language models.

Puyuan Peng, Brian Yan, Shinji Watanabe, and David Harwath. 2023. Prompting the hidden talent of web-scale speech models for zero-shot task generalization.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models.

Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2022. Do users write more insecure code with ai assistants?

Denis Peskoff and Brandon M Stewart. 2023. Credible without credit: Domain experts assess generative language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 427–438.

Denis Peskoff, Adam Visokay, Sander Schulhoff, Benjamin Wachspress, Alan Blinder, and Brandon M Stewart. 2023. Gpt deciphering fedspeak: Quantifying dissent among hawks and doves. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6529–6539.

Denis Peskov, Viktor Hangya, Jordan Boyd-Graber, and Alexander Fraser. 2021. Adapting entities across languages and cultures. *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Pouya Pezeshkpour and Estevam Hruschka. 2023. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*.

Carol W. Pfaff. 1979. Constraints on language mixing: Intrasentential code-switching and borrowing in spanish/english. *Language*, pages 291–318.

Jonathan Pilault, Xavier Garcia, Arthur Bražinskas, and Orhan Firat. 2023. Interactive-chain-prompting: Ambiguity resolution for crosslingual conditional generation with interaction.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion.

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en español: Toward a typology of code-switching. *Linguistics*, 18(7-8):581–618.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Preamble. 2024. Our product.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search.

Ratish Puduppully, Anoop Kunchukuttan, Raj Dabre, Ai Ti Aw, and Nancy F. Chen. 2023. Decomposed prompting for machine translation between related languages using large language models.

Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, Ming-Jie Ma, Pu Zhao, Si Qin, Xiaoting Qin, Chao Du, Yong Xu, Qingwei Lin, S. Rajmohan, and Dongmei Zhang. 2023. Taskweaver: A code-first agent framework. *ArXiv*, abs/2311.17541.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Reasoning with language model prompting: A survey.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023a. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shi Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bo Li, Ziwei Tang, Jing Yi, Yu Zhu, Zhenning Dai, Lan Yan, Xin Cong, Ya-Ting Lu, Weilin Zhao, Yuxiang Huang, Jun-Han Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023b. Tool learning with foundation models. *ArXiv*, abs/2304.08354.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019a. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019b. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sudha Rao and Hal Daumé III. 2019. Answer-based adversarial training for generating clarification questions. *arXiv preprint arXiv:1904.02281*.

Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. *arXiv*.

Philip Resnik, April Foreman, Michelle Kuchuk, Katherine Musacchio Schafer, and Beau Pinkham. 2021. Naturally occurring language as a source of evidence in suicide prevention. *Suicide and Life-Threatening Behavior*, 51(1):88–96.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21. ACM.

Megan L Rogers, Carol Chu, and Thomas Joiner. 2019. The necessity, validity, and clinical utility of a new diagnostic entity: Acute suicidal affective disturbance (asad). *Journal of Clinical Psychology*, 75(6):999.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models.

Shamik Roy, Raphael Shu, Nikolaos Pappas, Elman Mansimov, Yi Zhang, Saab Mansour, and Dan Roth. 2023. Conversation style transfer using few-shot learning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 119–143, Nusa Dua, Bali. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Runway. 2023. Gen-2 prompt tips. `https://help.runwayml.com/hc/en-us/articles/17329337959699-Gen-2-Prompt-Tips`.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications.

Gustavo Sandoval, Hammond Pearce, Teo Nys, Ramesh Karri, Siddharth Garg, and Brendan Dolan-Gavitt. 2022. Lost at c: A user study on the security implications of large language model code assistants.

Shubhra Kanti Karmaker Santu and Dongji Feng. 2023. Teler: A general taxonomy of llm prompts for benchmarking complex tasks.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *ArXiv*, abs/2009.07118.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics.

Douglas C. Schmidt, Jesse Spencer-Smith, Quchen Fu, and Jules White. 2023. Cataloging prompt patterns to enhance the discipline of prompt engineering. *Dept. of Computer Science, Vanderbilt University*. Email: douglas.c.schmidt, jesse.spencer-smith, quchen.fu, jules.white@vanderbilt.edu.

Allison Schuck, Raffaella Calati, Shira Barzilay, Sarah Bloch-Elkouby, and Igor I. Galynker. 2019a. Suicide crisis syndrome: A review of supporting evidence for a new suicide-specific diagnosis. *Behavioral sciences & the law*, 37 3:223–239.

Allison Schuck, Raffaella Calati, Shira Barzilay, Sarah Bloch-Elkouby, and Igor Galynker. 2019b. Suicide crisis syndrome: A review of supporting evidence for a new suicide-specific diagnosis. *Behavioral sciences and the law*, 37(3):223–239.

Sander Schulhoff. 2022. Learn Prompting.

Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Kost, Christopher Carnahan, and Jordan Boyd-Graber. 2023. Ignore this title and HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global prompt hacking competition. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4945–4977, Singapore. Association for Computational Linguistics.

Sander V Schulhoff. 2024. Prompt injection vs jailbreaking: What is the difference?

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023a. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023b. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting.

Harsha-Nori Scott Lundberg, Marco Tulio Correia Ribeiro. 2023. guidance. GitHub repository.

John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. 2023. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*.

Yongliang Shen, Kaitao Song, Xu Tan, Dong Sheng Li, Weiming Lu, and Yue Ting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *ArXiv*, abs/2303.17580.

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. 2022. Language models are multilingual chain-of-thought reasoners.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020a. Eliciting knowledge from language models using automatically generated prompts. *ArXiv*, abs/2010.15980.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020b. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Han-Chin Shing, Suraj Nair, Ayah Zirikly, Meir Friedenberg, Hal Daumé III, and Philip Resnik. 2018. Expert, crowdsourced, and machine assessment of suicide risk via online postings. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 25–36, New Orleans, LA. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Chenglei Si, Dan Friedman, Nitish Joshi, Shi Feng, Danqi Chen, and He He. 2023a. Measuring inductive biases of in-context learning with underspecified demonstrations. In *Association for Computational Linguistics (ACL)*.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2023b. Prompting gpt-3 to be reliable. In *International Conference on Learning Representations (ICLR)*.

Chenglei Si, Navita Goyal, Sherry Tongshuang Wu, Chen Zhao, Shi Feng, Hal Daumé III, and Jordan Boyd-Graber. 2023c. Large language models help humans verify truthfulness–except when they are convincingly wrong. *arXiv preprint arXiv:2310.12558*.

Chenglei Si, Weijia Shi, Chen Zhao, Luke Zettlemoyer, and Jordan Lee Boyd-Graber. 2023d. Getting MoRE out of Mixture of language model Reasoning Experts. *Findings of Empirical Methods in Natural Language Processing*.

Suzanna Sia and Kevin Duh. 2023. In-context learning as maintaining coherency: A study of on-the-fly machine translation using large language models.

Significant Gravitas. 2023. AutoGPT.

Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, and Yaniv Taigman. 2023. Text-to-4d dynamic scene generation.

Taylor Sorensen, Joshua Robinson, Christopher Rytting, Alexander Shaw, Kyle Rogers, Alexia Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. 2022. An information-theoretic approach to prompt engineering without ground truth labels. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 819–862, Dublin, Ireland. Association for Computational Linguistics.

Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. 2023. Evaluation metrics in the era of gpt-4: Reliably evaluating large language models on sequence to sequence tasks. *arXiv preprint arXiv:2310.13800*.

Michal Štefánik and Marek Kadlčík. 2023. Can in-context learners learn a reasoning concept from demonstrations? In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 107–115, Toronto, Canada. Association for Computational Linguistics.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2022. Selective annotation makes language models better few-shot learners.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models.

Lv Tang, Peng-Tao Jiang, Hao-Ke Xiao, and Bo Li. 2023. Towards training-free open-world segmentation via image prompting foundation models.

Eshaan Tanwar, Subhabrata Dutta, Manish Borthakur, and Tanmoy Chakraborty. 2023. Multilingual LLMs are better cross-lingual in-context learners with alignment. In *Proceedings of the 61st Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 6292–6307, Toronto, Canada. Association for Computational Linguistics.

Ming Tao, Hao Tang, Fei Wu, Xiao-Yuan Jing, Bing-Kun Bao, and Changsheng Xu. 2022. Df-gan: A simple and effective baseline for text-to-image synthesis.

Charlotte Thompson and Tiana Kelly. 2023. When hallucinations become reality: An exploration of ai package hallucination attacks. Darktrace Blog.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, Singapore. Association for Computational Linguistics.

Lindia Tjuatja, Valerie Chen, Tongshuang Wu, Ameet Talwalkwar, and Graham Neubig. 2024. Do llms exhibit human-like response biases? a case study in survey design. *Transactions of the Association for Computational Linguistics*, 12:1011–1026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. 2023. Gymnasium.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Rasul Tutunov, Antoine Grosnit, Juliusz Ziomek, Jun Wang, and Haitham Bou-Ammar. 2023. Why can large language models generate correct chain-of-thoughts?

Shubham Vatsal and Harsh Dubey. 2024. A survey of prompt engineering methods in large language models for different nlp tasks.

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. *arXiv preprint arXiv:2401.06766*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Conference on Empirical Methods in Natural Language Processing*.

Xingchen Wan, Ruoxi Sun, Hanjun Dai, Sercan O. Arik, and Tomas Pfister. 2023a. Better zero-shot reasoning with self-adaptive prompting.

Xingchen Wan, Ruoxi Sun, Hootan Nakhost, Hanjun Dai, Julian Martin Eisenschlos, Sercan O. Arik, and Tomas Pfister. 2023b. Universal self-adaptive prompting.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023b. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.

Jiaqi Wang, Zhengliang Liu, Lin Zhao, Zihao Wu, Chong Ma, Sigang Yu, Haixing Dai, Qiushi Yang, Yiheng Liu, Songyao Zhang, Enze Shi, Yi Pan, Tuo Zhang, Dajiang Zhu, Xiang Li, Xi Jiang, Bao Ge, Yixuan Yuan, Dinggang Shen, Tianming Liu, and Shu Zhang. 2023c. Review of large vision models and visual prompt engineering.

Jiaqi Wang, Enze Shi, Sigang Yu, Zihao Wu, Chong Ma, Haixing Dai, Qiushi Yang, Yanqing Kang, Jinru Wu, Huawen Hu, Chenxi Yue, Haiyang Zhang, Yiheng Liu, Xiang Li, Bao Ge, Dajiang Zhu, Yixuan Yuan, Dinggang Shen, Tianming Liu, and Shu Zhang. 2023d. Prompt engineering for healthcare: Methodologies and applications.

Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2023e. Software testing with large language model: Survey, landscape, and vision.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023f. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models.

Siyin Wang, Chao-Han Huck Yang, Ji Wu, and Chao Zhang. 2023g. Can whisper perform speech-based in-context learning.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023h. Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning.

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

Yaqing Wang, Jiepu Jiang, Mingyang Zhang, Cheng Li, Yi Liang, Qiaozhu Mei, and Michael Bendersky. 2023i. Automated evaluation of personalized text generation using large language models. *arXiv preprint arXiv:2310.11593*.

Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. 2019. Generalizing from a few examples: A survey on few-shot learning.

Yuqing Wang and Yun Zhao. 2024. Metacognitive prompting improves understanding in large language models.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhu Chen, Jie Fu, and Junran Peng. 2023j. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models.

Zhendong Wang, Yifan Jiang, Yadong Lu, Yelong Shen, Pengcheng He, Weizhu Chen, Zhangyang Wang, and Mingyuan Zhou. 2023k. In-context learning unlocked for diffusion models.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023l. Unleashing cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023a. Chain-of-thought prompting elicits reasoning in large language models.

Jerry Wei, Da Huang, Yifeng Lu, Denny Zhou, and Quoc V Le. 2023b. Simple synthetic data reduces sycophancy in large language models. *arXiv preprint arXiv:2308.03958*.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023c. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2022. Large language models are better reasoners with self-verification.

Jason Weston and Sainbayar Sukhbaatar. 2023. System 2 attention (is something you might need too).

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

Alex Wilf, Sihyun Shawn Lee, Paul Pu Liang, and Louis-Philippe Morency. 2023. Think twice: Perspective-taking improves large language models' theory-of-mind capabilities.

Simon Willison. 2022. Prompt injection attacks against gpt-3.

Simon Willison. 2024. Prompt injection and jailbreaking are not the same thing.

Genta Indra Winata, Liang-Kang Huang, Soumya Vadlamannati, and Yash Chandarana. 2023. Multilingual few-shot learning via language model retrieval.

Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. 2023a. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation.

Ning Wu, Ming Gong, Linjun Shou, Shining Liang, and Daxin Jiang. 2023b. Large language models are diverse role-players for summarization evaluation. *arXiv preprint arXiv:2303.15078*.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. *CHI Conference on Human Factors in Computing Systems*.

Xiaodong Wu, Ran Duan, and Jianbing Ni. 2023c. Unveiling security, privacy, and ethical concerns of chatgpt. *Journal of Information and Intelligence*.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. Fofo: A benchmark to evaluate llms' format-following capability.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023a. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023b. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, and Jian guang Lou. 2023. Re-reading improves reasoning in language models.

Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. 2023. Rcot: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023a. Large language models as optimizers.

Haibo Yang, Yang Chen, Yingwei Pan, Ting Yao, Zhineng Chen, and Tao Mei. 2023b. 3dstyle-diffusion: Pursuing fine-grained text-driven 3d stylization with 2d diffusion models.

Hui Yang, Sifu Yue, and Yunzhong He. 2023c. Auto-gpt for online decision making: Benchmarks and additional opinions.

Xinyi Yang, Runzhe Zhan, Derek F. Wong, Junchao Wu, and Lidia S. Chao. 2023d. Human-in-the-loop machine translation with large language model. In *Proceedings of Machine Translation Summit XIX Vol. 2: Users Track*, pages 88–98, Macau SAR, China. Machine Translation Summit.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023e. The dawn of lmms: Preliminary explorations with gpt-4v(ision). *ArXiv*, abs/2309.17421.

Binwei Yao, Ming Jiang, Diyi Yang, and Junjie Hu. 2023a. Empowering llm-based machine translation with cultural awareness.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023b. Tree of thoughts: Deliberate problem solving with large language models.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models.

Yao Yao, Zuchao Li, and Hai Zhao. 2023c. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. 2023. Large language models as analogical reasoners.

Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. 2023. Prompt engineering a prompt engineer.

Xi Ye and Greg Durrett. 2023. Explanation selection using unlabeled data for chain-of-thought prompting.

Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang goo Lee, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought.

Adeel Yousaf, Muzammal Naseer, Salman Khan, Fahad Shahbaz Khan, and Mubarak Shah. 2023. Video-prompter: an ensemble of foundational models for zero-shot video understanding.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large language model as attributed training data generator: A tale of diversity and bias. *arXiv preprint arXiv:2306.15895*.

Xiang Yue, Boshi Wang, Kai Zhang, Ziru Chen, Yu Su, and Huan Sun. 2023. Automatic evaluation of attribution by large language models. *arXiv preprint arXiv:2305.06311*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *arXiv preprint arXiv:2310.07641*.

Michael JQ Zhang and Eunsol Choi. 2023. Clarify when necessary: Resolving ambiguity through interaction with lms. *arXiv preprint arXiv:2311.09469*.

Quanjun Zhang, Tongke Zhang, Juan Zhai, Chunrong Fang, Bowen Yu, Weisong Sun, and Zhenyu Chen. 2023a. A critical review of large language model on software engineering: An example from chatgpt and automated program repair.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023b. Cumulative reasoning with large language models.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022a. Active example selection for in-context learning.

Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, and Hai Zhao. 2023c. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023d. Multimodal chain-of-thought reasoning in language models.

Ruochen Zhao, Xingxuan Li, Shafiq Joty, Chengwei Qin, and Lidong Bing. 2023a. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5823–5840, Toronto, Canada. Association for Computational Linguistics.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021a. Calibrate before use: Improving few-shot performance of language models.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023b. Large language models are effective table-to-text generators, evaluators, and feedback providers. *arXiv preprint arXiv:2305.14987*.

Yuyang Zhao, Zhiwen Yan, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. 2023c. Animate124: Animating one image to 4d dynamic scene.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023a. On large language models' selection bias in multi-choice questions. *arXiv preprint arXiv:2309.03882*.

Ge Zheng, Bin Yang, Jiajin Tang, Hong-Yu Zhou, and Sibei Yang. 2023b. Ddcot: Duty-distinct chain-of-thought prompting for multimodal reasoning in language models.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2023c. Take a step back: Evoking reasoning via abstraction in large language models.

Mingqian Zheng, Jiaxin Pei, and David Jurgens. 2023d. Is "a helpful assistant" the best role for large language models? a systematic evaluation of social roles in system prompts.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022a. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022b. Large language models are human-level prompt engineers.

Yucheng Zhou, Xiubo Geng, Tao Shen, Chongyang Tao, Guodong Long, Jian-Guang Lou, and Jianbing Shen. 2023. Thread of thought unraveling chaotic contexts.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory.

Zhichao Zuo, Zhao Zhang, Yan Luo, Yang Zhao, Haijun Zhang, Yi Yang, and Meng Wang. 2023. Cut-and-paste: Subject-driven video editing with attention control.

A Appendices

A.1 Definitions of Prompting

Reference	Prompt	Prompt Engineering
(Meskó, 2023)		The practice of designing, refining, and implementing prompts or instructions that guide the output of LLMs to help in various tasks. It is essentially the practice of effectively interacting with AI systems to optimize their benefits.
(Chen et al., 2023a)	the input of the model	the process of structuring input text for LLMs and is a technique integral to optimizing the efficacy of LLMs
(Santu and Feng, 2023)	refers to a textual input provided to the LLMs with the intention of guiding its output toward a specific task	involves crafting and revising the query or context in such a way that it elicits the desired response or behavior from LLMs
(Wang et al., 2023d)		involves designing effective prompts to guide the pre-trained language model in downstream tasks.
(Wang et al., 2023c)		the process of designing prompts that enable the model to adapt and generalize to different tasks. downstream tasks.
(Hou et al., 2023)	manually predefined natural language instructions	the careful design of specialized prompts
(Wang et al., 2023e)	input of the LLMs	communicate with LLMs to steer its behavior for desired outcomes
(White et al., 2023)	Instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities (and quantities) of generated output. Prompts are also a form of programming that can customize the outputs and interactions with an LLM. A prompt is a set of instructions provided to an LLM that programs the LLM by customizing it and/or en- hancing or refining its capabilities	an increasingly important skill set needed to converse effectively with large language models (LLMs), such as ChatGPT the means by which LLMs are programmed via prompts
(Heston and Khun, 2023)	the input	structuring the input in a specialized manner
(Liu et al., 2023b)		choosing a proper prompt the process of creating a prompting function $f_{prompt}(x)$ that results in the most effective performance on the downstream task.

(Hadi et al., 2023)	the instructions provided to an LLM to make it follow specified rules, automation of processes and to ensure that the output generated is of a specific quality or quantity	refers to the designing and wording of prompts given to LLMs so as to get a desired response from them.
(Neagu, 2023)		entails various strate- gies, including explicit instruction, and implicit context [21]. Explicit instruction involves providing explicit guidance or constraints to the model through instructions, examples, or specifications. Implicit context leverages the model's under- standing of the preceding context to influence its response
(Dang et al., 2022)		the systematic practice of constructing prompts to improve the generated output of a generative model

Table A.1: Definitions of Prompt and Prompt Engineering from different papers.

A.2 Extended Vocabulary

A.2.1 Prompting Terms

Context Window The context window is the space of tokens (for LLMs) which the model can process. It has a maximal length (the context length).

Priming (Schulhoff, 2022) refers to giving a model an initial prompt that lays out certain instructions for the rest of a conversation. This priming prompt might contains a role or other instructions on how to interact with the user. Priming can either be done in the system or user prompt (see below).

A.2.2 Prompt Engineering Terms

Conversational Prompt Engineering is Prompt Engineering *in colloquio*. That is, during the course of a conversation with a GenAI, a user may ask the GenAI to refine its output. In contrast, prompt engineering is often done by sending the GenAI a completely new prompt rather than continuing a conversation.

A.2.3 Fine-Tuning Terms

Prompt-Based Learning (Liu et al., 2023b), also known as Prompt Learning (Liu et al., 2023b; Wang et al., 2023d) refers to the process of using prompting-related techniques. It often is used in the context of fine-tuning, especially fine-tuning prompts. Due to conflicting usage, we do not use this term.

Prompt Tuning (Lester et al., 2021) refers to directly optimizing the weights of the prompt itself, usually through some form of gradient-based updates. It has also been referred to has Prompt Fine-Tuning. It should *not* be used to refer to discrete prompt engineering.

A.2.4 Orthogonal Prompt Types

We now discuss terminology for high-level ways of classifying prompts.

A.2.4.1 Originator

User Prompt This is the type of prompt that comes from the user. This is the most common form of prompting and is how prompts are usually delivered in consumer applications.

Assistant Prompt This "prompt" is simply the output of the LLM itself. It can be considered a prompt (or part of one) when it is fed back into the model, for example as part of a conversation history with a user.

System Prompt This prompt is used to give LLMs high level instructions for interacting with users. Not all models have this.

A.2.4.2 Hard vs Soft Prompts

Hard (discrete) Prompt These prompts only contain tokens that directly correspond to words in the LLM vocabulary.

Soft (continuous) Prompt These prompts contain tokens that may not correspond to any word in the vocabulary (Lester et al., 2021; Wang et al., 2023c). Soft prompts can be used when fine-tuning is desired, but modifying the weights of the full model is prohibitively expensive. Thus, a frozen model can be used while allowing gradients to flow through the prompt tokens.

$$\text{Hard Prompts} \subseteq \text{Soft Prompts}$$

A.2.4.3 Prediction Styles

In LLMs, a prediction style is the format in which it predicts the next token. There are two common formats for this in prompting research. We do not discuss non-text prediction styles.

Cloze In Cloze prompts, the token(s) to be predicted are presented as "slots to fill", usually somewhere in the middle of the prompt (Liu et al., 2023b). This is usually the case for earlier transformer models such as BERT (Chu and Lin, 2023).

Prefix In Prefix prompts, the token to be predicted is at the end of the prompt (Liu et al., 2023b). This is usually the case with modern GPT-style models (Radford et al., 2019b).

A.3 Datasheet

We present a datasheet (Gebru et al., 2021) with more information about the associated paper dataset, which is hosted on HuggingFace.

A.3.1 Motivation

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description.
This dataset was created to gather existing literature on prompt engineering in order to analyze all current hard prefix prompting techniques.

Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)?
This research was associated with the University of Maryland, Learn Prompting, and sponsored by OpenAI, but not created on the behalf of any particular organization.

Who funded the creation of the dataset? If there is an associated grant, please provide the name of the grantor and the grant name and number.
OpenAI contributed $10,000 in credits for their API.

A.3.2 Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description.
The dataset contains 1,565 research papers in PDF format. Any duplicate papers were removed automatically, though some could exist.

What data does each instance consist of? "Raw" data (e.g., unprocessed text or images) or features? In either case, please provide a description.
Each data instance is a research paper as a PDF.

Is there a label or target associated with each instance? If so, please provide a description.
No

Is any information missing from individual instances? If so, please provide a description, explaining why this information is missing (e.g., because it was unavailable). This does not include intentionally removed information, but might include, e.g., redacted text.
No.

Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description.
The papers were gathered in a semi-automated process which introduced the possibility of irrelevant papers being collected and relevant papers not being collected. There were manual reviews done for both possible errors to mitigate these errors.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)?
It is self-contained.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor–patient confidentiality, data that includes the content of individuals' non-public communications)? If so, please provide a description.
No.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? If so, please describe why.
The dataset contains some papers on prompt injection. These papers may contain offensive content including racism and sexism.

A.3.3 Collection Process

How was the data associated with each instance acquired?
The dataset was compiled from Arxiv, Semantic Scholar, and ACL.

What mechanisms or procedures were used to collect the data?
We wrote scripts to automatically query the APIs of Arxiv and Semantic Scholar.

Over what timeframe was the data collected?
The dataset was curated the duration of the research paper, primarily in February of 2024.

Were any ethical review processes conducted?
No.

A.3.4 Preprocessing/ Cleaning/ Labeling

Was any preprocessing/cleaning/labeling of the data done?
After collecting data from different sources, we removed duplicate papers and did a manual and semi-automated review of papers to ensure they were all relevant.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data?
No, we do not anticipate the use of our preprocessed data. However, raw data can be recovered from the links we store.

Is the software that was used to preprocess/clean/label the data available?
It is contained within our code repository on Github.

A.3.5 Uses

Has the dataset been used for any tasks already?
No.

Is there a repository that links to any or all papers or systems that use the dataset?
Yes.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?
All of the papers we collected were written in English. It is possible some papers were not included due to a translation not being available.

Are there tasks for which the dataset should not be used?
No.

A.3.6 Distribution

Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created?
No.

A.3.7 Maintenance

Who will be supporting/hosting/maintaining the dataset?
Our team will continue maintenance.

How can the owner/curator/manager of the dataset be contacted?
Please email us at sanderschulhoff@gmail.com

Is there an erratum?
No.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?
Yes, anyone is free to use/modify the data.

A.4 Keywords

Here are the keywords we used for search.

- jailbreak prompt
- prompt an llm
- prompt a large language model
- prompt injection
- prompt optimization
- prompt engineering
- few-shot learning
- few shot learning
- prompt-based methods
- prompt based methods
- prompting-based methods
- prompting based methods
- few-shot prompt
- few shot prompt
- one-shot prompt
- one shot prompt
- few-shot prompting
- few shot prompting
- one-shot prompting
- one shot prompting
- prompting techniques
- prompt engineering techniques
- llm prompting
- large language model prompting
- 0-shot prompt
- 0 shot prompt
- zero-shot prompt
- many-shot prompt
- zero-shot prompting
- many-shot prompting

- in-context learning
- in context learning
- transformer model prompts
- prompt-based transfer learning
- nlp prompting strategies
- llm interpretability via prompts
- curriculum learning with prompts
- feedback loops in llm prompting
- human-in-the-loop prompting
- token-efficient prompting
- multimodal prompting
- instruction prompting
- prompt templating
- prompt template

A.5 Prompt for Systematic Literature Review

Please find the prompt we used here. We present it in text in this document, but note that you should use the version in our codebase rather than copy and paste this.

We used the following system prompt:

You are a lab assistant, helping with a systematic review on prompt engineering. You've been asked to rate the relevance of a paper to the topic of prompt engineering. To be clear, this review will strictly cover hard prefix prompts. For clarification: Hard prompts have tokens that correspond directly to words in the vocab. For example, you could make up a new token by adding two together. This would no longer correspond to any word in the vocabulary, and would be a soft prompt Prefix prompts are prompts used for most modern transformers, where the model predicts the words after this prompt. In earlier models, such as BERT, models could predict words (e.g. <MASK>) in the middle of the prompt. Your job is to be able to tell whether a paper is related to (or simply contains) hard prefix prompting or prompt engineering. Please note that a paper might not spell out that it is using "hard prefix" prompting and so it might just say prompting. In this case, you should still rate it as relevant to the topic of prompt engineering. Please also note, that a paper that focuses on training a model as opposed to post-training prompting techniques is considered irrelevant. Provide a response in JSON format with two fields: 'reasoning' (a single sentence that justifies your reasoning) and 'rating' (a string that is one of the following categories: 'highly relevant', 'somewhat relevant', 'neutrally relevant', 'somewhat irrelevant', 'highly irrelevant') indicating relevance to the topic of prompt engineering)

Then, we used this user prompt template to input information for each paper:

Title: '{title}', Abstract: '{abstract}'. Rate its relevance to the topic of prompt engineering as one of the following categories: 'highly relevant', 'somewhat relevant', 'neutrally relevant', 'somewhat irrelevant', 'highly irrelevant', and provide text from the abstract that justifies your reasoning

A.6 Evaluation Table

ID	Model	Prompt				Output Space	Type	Res.	Batch
		Roles	CoT	Definition	Few-Shot				
(Kocmi and Federmann, 2023b)	GPT-family					DA, sMQM, stars, classes	E	S	
(Lu et al., 2023c)	Dav3, GPT-4-Turbo, GPT-4		✓	✓	✓	Error Span → Score	E	S	✓
(Fernandes et al., 2023)	PaLM		✓	✓	✓	Error Span	I	S	
(Kocmi and Federmann, 2023a)	GPT-4		✓	✓	✓	Error Span	I	S	✓
(Araújo and Aguiar, 2023)	ChatGPT			✓		Likert [1-5]	E	S	✓
(Wang et al., 2023b)	ChatGPT			✓		DA, stars	E	S	
(Liu et al., 2023d)†	GPT-3.5, GPT-4			✓		Likert [1-10]	I	M	
(Chan et al., 2024)	ChatGPT, GPT-4	✓	✓			Likert [1-10]	I	M	
(Luo et al., 2023)	ChatGPT		✓	✓		yes/no;A/B; Likert [1-10]	E	S	
(Hada et al., 2024)	GPT-4-32K			✓	✓	[0,1,2] or binary	E	S	✓
(Fu et al., 2023a)	GPT-3, OPT, FLAN-T5, GPT-2					Probability	I	S	
(Gao et al., 2023c)	ChatGPT			✓		Likert [1-5], Pairwise, Pyramid, 0/1	E	S	
(Chen et al., 2023g)	ChatGPT					Likert [1-10]; yes/no; pairwise: A/B/C	E & I	S	
(He et al., 2023a)	GPT-4			✓		Likert [1-5]	E	S	
(Sottana et al., 2023)	GPT-4			✓		Likert [1-5]	E	S	
(Chen et al., 2023c)	GPT, Flan-T5		✓			Yes/No	E	S	
(Zhao et al., 2023b)	GPT-3.5, GPT-4		✓		✓	true/false	E	S	
(Wu et al., 2023b)	GPT-3	✓				pairwise voting	E	M	✓
(Wang et al., 2023i)	PaLM 2-IT-L					A/B	E	M	
(Jia et al., 2023)	LLaMa7b					Probability	I	S	
(Yue et al., 2023)	ChatGPT, Alpaca, Vicuna, GPT-4			✓	✓	Yes/No	E	S	
(Li et al., 2023e)	GPT-3.5, GPT-4, Bard, Vicuna		✓			Pairwise	I	M	
(Liu et al., 2023f)	ChatGPT, Vicuna, chatGLM, StableLM			✓		continuous [0-1]	E	S	
(Bai et al., 2023b)	GPT-4, Claude, ChatGPT, Bard, Vicuna			✓		Likert [1-5]	E	S	
(Dubois et al., 2023)	GPT-4, ChatGPT, Dav3			✓	✓	pairwise	E	M	✓
(Liu et al., 2023h)†	GPT-4-32K			✓		Likert [1-5]	E	S	
(Wang et al., 2023h)	GPT-4-Turbo, ChatGPT, GPT-4, Vicuna		✓			Likert [1-10]	E	M	
(Zeng et al., 2023)	GPT-4, ChatGPT, LLaMA-2-Chat, PaLM2, Falcon	✓	✓		✓	Pairwise	E	S	
(Zheng et al., 2023b)	Claude-v1, GPT-3.5, GPT-4		✓		✓	Pairwise/Likert [1-10]	E	S/M	
(Lin and Chen, 2023)	Claude-v1.3					Likert [0-5], Likert [0-100]	E	S	✓

Table A.2: Evaluation Paper Summary. E: Explicit (whether the model generates an assessment), I: Implicit (whether an assessment is derived from the model output); Response (Res.) S: Single response, M: Multiple responses; †: Model generated instruction;

A.7 Entrapment Prompting Process

This section contains the thought process of our prompt engineer as he developed the prompt.

A.7.1 Exploration

- First did a bit of dataset exploration, looking at length/label distribution, then chose entrapment to start with.
- Checked if gpt-1106-preview understood entrapment, by asking it to define the term WRT SCS. It did not.

A.7.2 Getting a Label

- Showed it the definition of entrapment in the system prompt and asked it to label a data point, but it responded saying I should seek mental health support.
- I put the instructions in the user prompt, but get a similar response.
- Appending "Is this entrapment? Yes or no" got it to actually respond with a label.
- I gave it a reject example, but it labeled it as accept, and also sometimes gives emotional responses about getting help.
- Using 10-shot prompting, it still gives the get help responses.
- I tried removing the definition, got the same problem. It seems that it doesn't like "accept"/"reject" as labels.
- I added back the definition, changed labels to entrapment/not entrapment, it often says "not enough info to determine".
- I switched to GPT-4-32K, which gave one word responses and correctly predicted a reject and an accept datapoint.

A.7.3 Varying Prompting Techniques

- Tested 0-shot with context.
- 10-shot with context, showed improvement.
- Taking first failing instance into playground. (12th in train file)
- Tried doing 'A: Let's go step by step and figure this out: ", output "A: Yes"
- Did "A: Let's go step by step and figure this out: ", OUTPUT:

> The individual in this post is expressing feelings of hopelessness, loneliness, and despair. They mention a significant loss (their mother) and a subsequent decline in their mental health, turning to alcohol as a coping mechanism. They also mention financial struggles and the possibility of becoming homeless. They express feelings of guilt and shame about their situation and compare themselves negatively to their friends and family. Most concerning is their detailed plan for suicide, involving sleeping pills and alcohol. They also mention considering stepping in front of a car. Given these factors, the label for this post would be "Yes" for expressing feelings of entrapment.

- "A: Before predicting a label, let's go step by step and write out your thought process. Use the items in the definition of entrapment to support your conclusion.", OUTPUT:

> Before predicting a label, let's go step by step and write out your thought process. Use the items in the definition of entrapment to support your conclusion.

- Began discussing with the AI "they don't ever say they feel trapped or that there is no way out, how did u get that idea?", OUTPUT:

> The individual doesn't explicitly state that they feel trapped or that there is no way out, but their narrative suggests a sense of entrapment. They mention that they are facing eviction, struggling with debt, and dealing with a decrease in income. They also express a sense of hopelessness and despair, and contemplate suicide as a potential solution to their problems. These elements suggest a feeling of being trapped in their current circumstances with no apparent way out. However, it's important to note that this interpretation is based on the information provided in the post and the individual's expressed feelings and thoughts.

- Asked it to help reformulate instructions to make explicitness more clearly needed, still predicts entrapment.
- Added "ENTRAPMENT MUST BE EXPLICIT, NOT IMPLICIT." to instruction at top after definitions (no CoT inducer)
- Also tried with previous CoT inducer, still says entrapment in a hard to parse out way.
- Added "If the person does not explicitly say they feel trapped, this doesn't count as entrapment." (no CoT), still entrapment.
- Talking to AI, it thinks "Today I found out I have 10 days to vacate my apartment or I'll be formally evicted. I'm 2 months behind on my rent due to a bad time where I got demoted at work and rent from making roughly $1000 ever 2 weeks to around $450. If I get evicted, I'll probably be homeless" is feeling trapped/stuck.
- Tried "Before stating a label, write out reasoning using the grid above about why this may/may not be entrapment:", still thinks entrapment.
- paste in email: [REDACTED] above definition of entrapment
- remove shots, says "Entrapment"
- add this after def: IMPORTANT: Only label the post as entrapment if they explicitly say that they feel trapped., says "Yes"
- In the prompt, gave it CoT reasoning. (18.txt), and tried with the next wrongly labeled one (15), (full prompt, 19.txt)
- Tested this on everything except first 20, did pretty well
- Tried removing email, performance dropped of a cliff
- At this point, I am thinking that giving examples with reasoning helps (obviously)
- Tried to add 10 shots in for free, before the last one with reasoning, bad results

A.7.3.1 AutoCoT

- Develop dataset using this prompt (22.txt). Then ask it "Why?". If it disagrees, I say "It is actually not entrapment, please explain why." (accidentally duplicated email 23.txt)
- Just for fun, tried 0 shot full context (had to adjust verbalizer)
- tried this with special verbalizer which catches "This post does not meet the criteria for Entrapment."
- Tested my generated data, beat 0.5 F1
- Doing 10 more exemplars w autocot. Sometimes responds immediately with reasoning like "This post does not meet the criteria for Entrapment as the individual does not explicitly express feelings of being trapped or hopeless.", so just use that if so. Sometimes get refusal "I'm really sorry to hear that you're feeling this way, but I'm unable to provide the help that you need. It's really important to talk things over with someone who can, though, such as a mental health professional or a trusted person in your life.", just ask "Explain why it is not entrapment." after if so.
- performance didnt really improve, realized about 11% are getting -1, meaning not extracted properly. Retrying with full words "Question" instead of Q, also for reasoning and answer.
- this led to higher inability to parse, at about 16%.

A.7.3.2 Developing Answer Extraction

- put first failing to parse one in (22), and developed a prompt for it.
- did worse: (0.42857142857142855, 0.5051546391752577, 0.8571428571428571, 0.2857142857142857)
- only using extracted label if have -1 helps slightly to (0.48, 0.61, 0.8571428571428571, 0.3333333333333333)
- going back to best performing prompt–10 QRA shot, and performing extraction with any -1s, doesnt help other than gently boosting accuracy, perhaps when it doesnt answer

A.7.3.3 Iterating on Email

- tried best perf, with no email
- tried with deduped email, worse results
- noticed that ones its unsure about often contained 1 labels that should be 0, so trying to "recover" these doesnt help
- try moving around exemplar order, performing extraction, didnt help
- triplicated email, didnt help

A.8 Formally Defining a Prompt

"Prompt" is a widely used term, but uses and definitions differ widely across research. As a result, it is difficult to create a formal, mathematical definition for a prompt. In this section, we outline some formalisms for prompt engineering.

As a conditioning Mechanism. Qiao et al. (2022) present the following definition, which involves the prompt $\mathcal{T}$ and a question $\mathcal{Q}$ as conditioning mechanisms on predicting the next token. Note that they appear to use Brown et al. (2020)'s original definition of prompt, which refers to the non-question part of the prompt (e.g. few-shot exemplars, instructions).

$$p(\mathcal{A} \mid \mathcal{T}, \mathcal{Q}) = \prod_{i=1}^{|\mathcal{A}|} p_{\text{LM}}\left(a_i \mid \mathcal{T}, \mathcal{Q}, a_{1:i-1}\right) \tag{A.1}$$

Here, the prompt and question condition the pre-trained LLM p_{LM}. The $a_{1:i-1}$ are previously generated answer tokens and $\mathcal{A}$ a complete answer.

Templating. The above formalization does not include the notion of maximizing a scoring or utility function (e.g. accuracy on a dataset), which prompts are often designed to do. Additionally, prompt engineers often seek to design prompt template rather than prompts. Here, we reformulate eq. (A.1) to include the prompt template:

$$p(\mathcal{A} \mid \mathcal{T}(x^*)) = \prod_{i=1}^{|\mathcal{A}|} p_{\text{LM}}\left(a_i \mid \mathcal{T}(x^*), a_{1:i-1}\right) \tag{A.2}$$

We replace $\mathcal{Q}$ with $x^* \in \mathcal{D}_{\text{eval}}$, an item from a dataset (e.g., evaluation data). Additionally, we replace $\mathcal{Q}$ on the right side with $\mathcal{T}(x)$. $\mathcal{T}(\cdot)$ is a prompt template: a function that accepts some item as input then returns a prompt that is used to condition the model.

Few-Shot Prompting. Often, an important part of the prompting process is the use of few-shot exemplars. $\mathcal{D}_{train}$ is training data (used to build the prompt) and $\mathcal{X}$ is a test set for evaluation.

$$\mathcal{D}_{\text{train}} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{A.3}$$

$$\mathcal{X} = \{x_1^*, x_2^*, ..., x_m^*\} \tag{A.4}$$

In the few-shot setting, the prompt template function $\mathcal{T}(\cdot)$ also takes as input one or more training samples $\mathcal{X} = \{(x_i, y_i)\}_1^n \subset \mathcal{D}_{\text{train}}$

$$p(\mathcal{A} \mid \mathcal{T}\left(\mathcal{X},\ x^*\right)) = \prod_{i=1}^{|\mathcal{A}|} p_{\text{LM}}\left(a_i \mid \mathcal{T}\left(\mathcal{X},\ x^*\right), a_{1:i-1}\right) \tag{A.5}$$

Optimization. As mentioned, it is often desirable to speak about improving prompts (prompt templates, that is) with respect to a scoring function, usually defined with respect to a dataset.

$$\mathcal{T}^* = \underset{\mathcal{T}}{\text{argmax}}\ \mathbb{E}_{x_i, y_i \sim \mathcal{D}}\left[S\left(p_{\text{LM}}(\mathcal{A}|\mathcal{T}(x_i)), y_i\right)\right] \tag{A.6}$$

In this definition, we are evaluating over a dataset $\mathcal{D}$ with respect to the scoring function $S(\cdot)$. $S(\cdot)$ evaluates the output $\mathcal{A}$, generated by the LLM conditioned on the prompt $\mathcal{T}(\S)$. y_i are labeled outputs that can be used by S.

In some cases, there may not be any labeled data y_i, and $S(\cdot)$ may be reference-free.

Other considerations. These formalisms could be adapted to cater to CoT, retrieval systems, and more. Here we describe a simple setup which is most descriptive of the prompting process without adding too much complexity.

We also draw attention to the lesser known concept of answer engineering. $E(\mathcal{A})$ is a transformation function over the raw LLM output that allows it to be compared to the ground truth.

$$\mathcal{A} \sim p_{\text{LM}}(\mathcal{A} \mid \mathcal{T}(x_i), y_i) \tag{A.7}$$

$$\mathcal{T}^* = \underset{\mathcal{T}, E}{\text{argmax}}\, \mathbb{E}_{x_i, y_i \sim \mathcal{D}} \left[S\left(E(\mathcal{A}), y_i \right) \right] \tag{A.8}$$

A.9 In-Context Learning Definitions Disambiguation

Brown et al. (2020) seemingly offer two different definitions for ICL. All bolding in this section is our own.

> Recent work [RWC+19] attempts to do this via what we call "in-context learning", using the text input of a pretrained language model as a form of task specification: the model is **conditioned on a natural language instruction and/or a few demonstrations of the task** and is then expected to complete further instances of the task simply by predicting what comes next.

However, they later appear to define it as few-shot only:

> For each task, we evaluate GPT-3 under 3 conditions: (a) **"few-shot learning", or in-context learning where we allow as many demonstrations as will fit into the model's context window** (typically 10 to 100), (b) "one-shot learning", where we allow only one demonstration, and (c) "zero-shot" learning, where no demonstrations are allowed and only an instruction in natural language is given to the model.

However, they include this image that clarifies the matter:

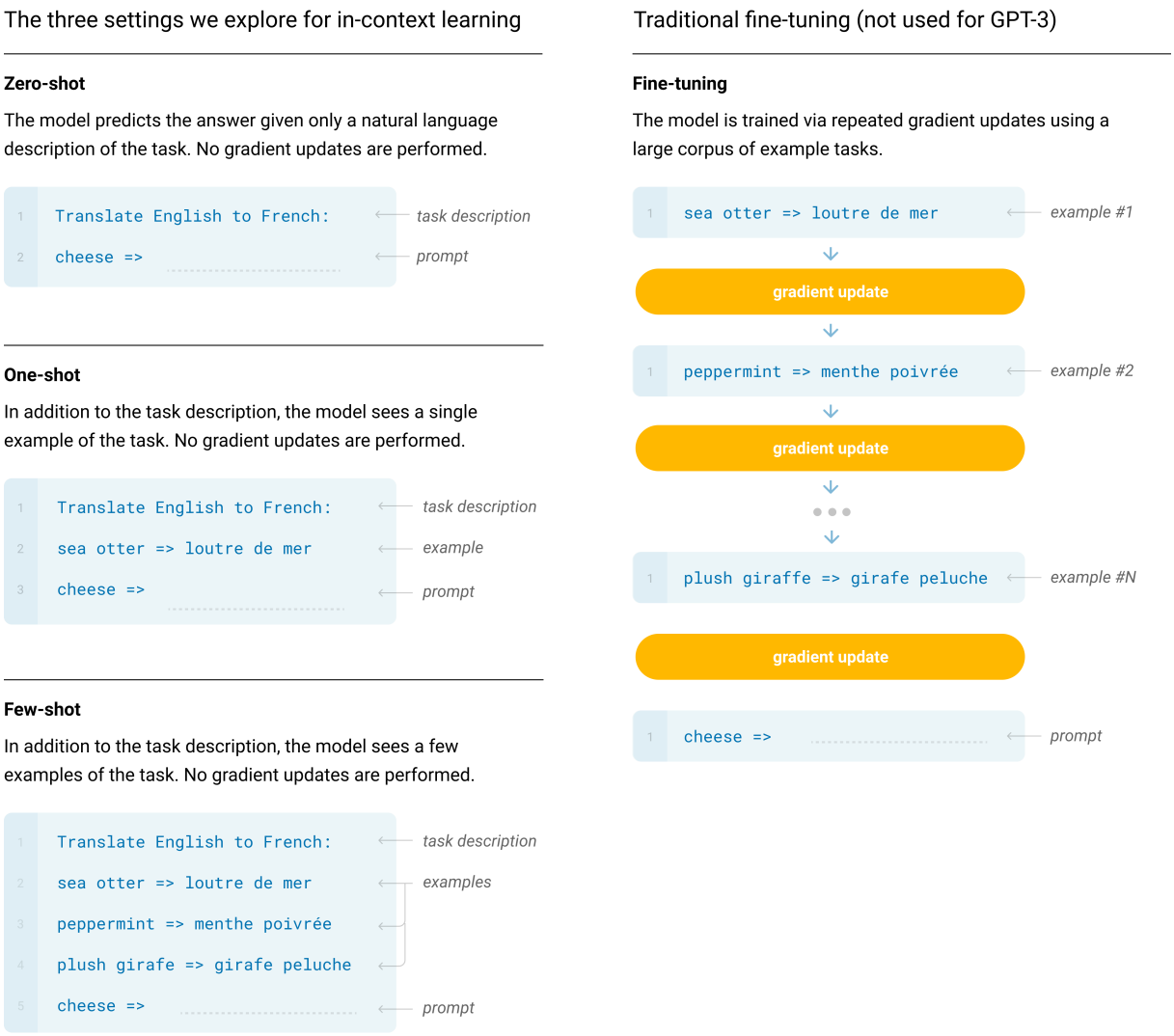

Figure A.1: ICL from Brown et al. (2020).

Additionally, they explicitly state that ICL does not necessarily involve learning new tasks.

> To avoid this confusion, we use the term "meta-learning" to capture the inner-loop / outer-loop structure of the general method, and the term "in context-learning" to refer to the inner loop of meta-learning. We further specialize the description to "zero-shot", "one-shot", or "few-shot" depending on how many demonstrations are provided at inference time. **These terms are intended to remain agnostic on the question of whether the model learns new tasks from scratch at inference time or simply recognizes patterns seen during training** – this is an important issue which we discuss later in the paper, but "meta-learning" is intended to encompass both possibilities, and simply describes the inner-outer loop structure.

We use Brown et al. (2020)'s broad definition, though note that practitioners often use ICL to refer to situations in which the model appears to be learning new tasks from the prompt. Our definition differs from Dong et al. (2023)'s formal definition, even though it is also derived from (Brown et al., 2020).

A.10 Contributions

The following are the contributions made by the team members in various sections of this paper. Most authors conducted reviews of other sections as well.

Advisors

- **Denis Peskoff:** Assisted with paper organization and final review.
- **Alexander Hoyle:** Provided guidance on writing, meta-analysis approach, and ran automated baselines for case study.
- **Shyamal Anadkat:** Assisted with the overall review of the paper and the etymology and definitions.
- **Jules White:** Built trees for technique taxonomies.
- **Marine Carpaut:** Framed, reviewed and suggested papers for the multilingual section.
- **Phillip Resnik:** Principal Investigator

SCS Labeling

- **Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker:** reviewed and gave advice for this section.

Benchmarking and Agents

- **Konstantine Kahadze:** Team leader for the Benchmarking section; managed MMLU benchmarking codebase, contributed to Security and Meta Analysis.
- **Ashay Srivastava:** Team leader for the Agents section, reviewed papers for human review, worked on the tool use agents section. Worked on the compilation of contributions.
- **Hevander Da Costa**: Contributed to the Benchmarking section and Meta Review datasets list, reviewed literature on LLM code generation and prompting techniques. Added literature review content to the Agents section.
- **Feileen Li:** Worked on the tool use agents section, assisted with the human paper review.

Alignment and Security

- **Nishant Balepur:** Team leader for the alignment section, helped with high-level discussions in benchmarking, and reviewed drafts.
- **Sevien Schulhoff:** Team leader for the security section and contributed to the benchmarking section.

Related Works and Section Contributions

- **Chenglei Si:** Suggested related works and edited section 2.2 and section 7.
- **Pranav Sandeep Dulepet:** Contributed definitions for section 2 and worked on segmentation and object detection in the multimodal section.
- **HyoJung Han:** Contributed to the Multimodal section, especially the speech+text part, and wrote the audio prompting section.
- **Hudson Tao:** Authored sections on image, video, and 3D within multimodal, reviewed papers for human review; maintained GitHub codebase, and built the project website.
- **Amanda Liu:** Authored taxonomic ontology sections, conducted background research for introduction and related work, developed code pipelines for meta-analysis graphs

- **Sweta Agrawal:** Team lead for evaluation section.
- **Saurav Vidyadhara:** Assisted with general review and revising taxonomy trees.
- **Chau Pham:** Assisted with meta review, including automated analysis of topics.

Multilingual Prompting and Meta Analysis

- **Dayeon Ki:** Led the Multilingual prompting section, conducted review on related papers, and wrote Section 3.1.
- **Yinheng Li:** Worked on section 2.2 text-based techniques, reviewed techniques, and contributed to drafting figure 2.2.
- **Saloni Gupta:** Wrote tests for paper compilation, helped set up paper pipeline, and worked on the code diagram and grammar for the paper.
- **Gerson Kroiz:** Involved with section 1.1 and defining a prompt.
- **Aayush Gupta:** Contributed to the Meta Analysis, compiling papers, and generating visualization graphs.
- **Michael Ilie:** Co-Lead Author, managed codebase, ran experiments, collected data, and helped with various sections including the PRISMA review figure and the SCS prompting case study.
- **Sander Schulhoff:** Lead Author