Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm

Laria Reynolds
moire@knc.ai
KNC.ai

Kyle McDonell
kyle@knc.ai
KNC.ai

ABSTRACT

Prevailing methods for mapping large generative language models to supervised tasks may fail to sufficiently probe models' novel capabilities. Using GPT-3 as a case study, we show that 0-shot prompts can significantly outperform few-shot prompts. We suggest that the function of few-shot examples in these cases is better described as locating an already learned task rather than meta-learning. This analysis motivates rethinking the role of prompts in controlling and evaluating powerful language models. We discuss methods of prompt programming, emphasizing the usefulness of considering prompts through the lens of natural language. We explore techniques for exploiting the capacity of narratives and cultural anchors to encode nuanced intentions and techniques for encouraging deconstruction of a problem into components before producing a verdict. Informed by this more encompassing theory of prompt programming, we also introduce the idea of a *metaprompt* that seeds the model to generate its own natural language prompts for a range of tasks. Finally, we discuss how these more general methods of interacting with language models can be incorporated into existing and future benchmarks and practical applications.

CCS CONCEPTS

• **Computing methodologies** → **Control methods**; **Natural language processing**; • **Human-centered computing** → **Natural language interfaces**.

KEYWORDS

language models, transformers, GPT-3, few-shot learning, prompt programming, metaprompts, serial reasoning, semiotics

ACM Reference Format:
Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '21 Extended Abstracts), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3411763.3451760

1 MOTIVATION

The recent rise of massive self-supervised language models such as GPT-3 [3] and their success on downstream tasks has brought us one step closer to the goal of task-agnostic artificial intelligence systems. However, despite the apparent power of such models, current methods of controlling them to perform specific tasks are extremely limited. In order to properly evaluate their capabilities and extract useful work from these models, new methods are required.

Prior to GPT-3, the standard approach to the evaluation and use of such models has involved fine-tuning on a portion of a task dataset [12]. GPT-3 achieved state-of-the-art performance on a wide variety of tasks without fine tuning, using only *few-shot* prompts, in which examples of solved tasks (shots) are provided as input to the trained model. However, while the few-shot format was sufficient to reveal surprising performance on these tasks, we argue that novel methods of prompting can be more effective than either fine-tuning or the few-shot format at extracting specific learned behaviors from self-supervised language models.

Contrary to the interpretation of implied by the title of the original GPT-3 paper by Brown et al. [3], *Language models are few-shot learners*, we argue that GPT-3 is often not actually *learning* the task during run time from few-shot examples. Rather than instruction, the method's primary function is *task location* in the model's existing space of learned tasks. This is evidenced by the effectiveness of alternative prompts which, with no examples, can elicit comparable or superior performance to the few-shot format.

This motivates new approaches which explicitly pursue the goal of task location. We propose exploring more general methods of prompt programming and specifically techniques for communicating task intention and structure to a self-supervised model in the modality it was trained: natural language. With a few caveats, we want to find prompts which we would expect a human to complete in a way that accomplishes the desired task.

In this work, we investigate the few-shot paradigm and find that its performance can be matched or exceeded by simple 0-shot prompts. We explore the nature of successful 0-shot prompts and propose general methods of prompt programming through the lens of natural language semiotics. We demonstrate novel prompts which force a language model to break a problem into components before producing a verdict, and we introduce the concept of *metaprompt programming*, an approach which offloads the job of writing a task-specific prompt to the language model itself. Finally, we discuss how these ideas can be incorporated into existing and future benchmarks to allow us to better probe the capabilities of large language models.

2 RELATED WORK

Recent work in the literature has focused on controlling natural language generation using traditional approaches from machine learning like novel architectures which condition outputs [15, 16], more advanced sampling techniques [6, 11], gradient-based

CHI '21 Extended Abstracts, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8095-9/21/05…$15.00
https://doi.org/10.1145/3411763.3451760

optimization of prompts [17, 22], and task-specific adapter networks [25]. See [24] for a survey of these recent methods. Past work has also explored improving the few-shot paradigm by dynamically selecting the most relevant examples for each task [9, 18].

In comparison, little work on natural-language approaches to prompt programming has been formalized. Instead, successful prompt programming techniques have primarily been shared on blogs and social media among users of OpenAI's API and AI Dungeon.

Due to the decentralized form that most explorations of prompt programming have taken, it is not feasible for us to to compile all relevant contributions here. We instead give the following brief, non-exhaustive survey of explorations which have gone beyond the few-shot paradigm.

G. Branwen has given the most comprehensive survey of GPT-3's capabilities through demonstrations of it writing fiction, poetry, and performing tasks like PDF cleaning. He has written extensively about his intuitions of working with GPT-3 and his methods of prompt programming [2]. Sabeti has written about the effect of the context provided by a prompt on writing quality [21]. Robertson has written about amplifying GPT-3's mathematical capabilities through a dialogue that guides it to break a problem into steps [20]. Twitter user KaryoKleptid has posted experiments along a similar vein, using dialogues to prompt GPT-3 (via AI Dungeon) to break problems into steps and follow procedures such as brute force checking [13, 14], achieving impressive results on math problems.

Our work synthesizes and expands on the methods pioneering by these explorations, representing a modest step towards formalizing effective natural language prompt programming techniques.

3 INVESTIGATING FEW-SHOT PROMPTING

GPT-3 was evaluated on tasks with 0, 1, and n-shot prompts (containing only a natural language description, one solved example, and n solved examples, respectively). GPT-3 consistently performs better when more examples are provided, with 0-shot performance often achieving less than half of the score of many-shot tests. A common interpretation of this result is that GPT-3 is learning from the examples at runtime [3]. However, these performance improvements can be alternatively interpreted as the result not of conventional learning, but of a mechanism we refer to as *task location*.

For example, for certain tasks, such as translation, a small number of samples is insufficient to learn anything substantial about the task. Instead, GPT-3 must rely primarily, if not entirely, on the knowledge of vocabulary and grammar of both the source and target languages embedded in its trained weights. Rather than viewing these tasks as *few-shot-learning*, we will explicitly show that these prompts primarily direct the model to access existing knowledge. We do so by investigating whether examples (training samples) are even necessary.

3.1 The success of 0-shot prompts

Due to budget constraints, we explore a single, illustrative example, a French-to-English translation task. We find that 0-shot prompts can match and even exceed standard few-shot performance. Our results in table 1 show that the 0-shot accuracy reported by Brown et al. [3] can be improved substantially with even minor prompt engineering. Most significantly, the extremely simple prompt in

Table 1: We report BLEU scores for variants of GPT-3 using different prompt formats on the WMT'14 Fr-En translation task [1] as measured by SacreBLEU [19]. First are results reported in the original GPT-3 paper [3] on the 6.7B and 13B parameter versions of GPT-3, our attempts to reproduce the results according to those exact specifications using the *Babbage* and *Curie* models available from OpenAI's API, and finally results from custom prompts described in (Figures 1,2). The difference in the reproduced results may be attributable to changes in the OpenAI API after the publication of their results. We were unable to replicate the 64-shot test due to API constraints and instead replaced it with a 10-shot test.

Prompt	Babbage / 6.7B	Curie / 13B
OpenAI 0-shot	15.5	22.4
OpenAI 1-shot	31.6	31.4
OpenAI 64-shot	36.4	38.3
Reproduced OpenAI 0-shot	15.9	18.7
Reproduced OpenAI 1-shot	21.8	24.1
Reproduced OpenAI 10-shot	25.1	27.9
Simple colon 0-shot	23.5	33.3
Simple colon 1-shot	18.0	27.6
Simple colon 10-shot	24.1	33.4
Master translator 0-shot	26.5	32.9

```
French: example-source-phrase
English: example-target-phrase

French: example-source-phrase
English: example-target-phrase

[...]

French: source-phrase
English:
```

Figure 1: The "Simple Colon" prompt format. For few-shot tasks, additional examples are provided as shown. Text in bold is to be replaced by source and target language text examples.

Figure 1, which includes only the names of the source and target languages, performs better than the 10-shot prompt in the style of the original GPT-3 paper.

We believe that this phenomena holds true for other tasks, and has caused the 0-shot or baseline performance of GPT-3 and similar models to be significantly underestimated. Gaining a better understanding of the capabilities of these large language models is of the utmost importance in controlling them effectively. The vast repertoire of functions contained within GPT-3 that do not need to be learned at runtime allows for great flexibility in 0-shot

```
A French phrase is provided: source-phrase
The masterful French translator flawlessly translates the phrase
into English:
```

Figure 2: The "Master Translator" prompt format. Text in bold is to be replaced by source and target language text examples.

prompting and encourages exploring more general methods of prompt programming.

3.2 Examples don't always help

In our experiment, the simple colon prompt (Figure 1) performed significantly worse with 1-shot compared to 0-shot. By examining the output of GPT-3 on this task, we found that the decreased performance was due to semantic contamination from the 1-shot example. Instead of treating examples as a *categorical* guide, it is inferred that their semantic meaning is relevant to the task, e.g. the examples are interpreted as part of a consecutive narrative. Indeed, we found this was true more generally of low-shot prompts across a variety of tasks. This effect of contamination from few-shot examples has been successfully used to improve the performance of GPT-3 by selecting in-context examples for each task [18].

4 PROMPT PROGRAMMING

Rewriting a prompt can result in significant changes to the performance of a language model on tasks. That motivates the question: Is there a methodology which we can follow to craft prompts more likely to yield desired behavior?

Prompt engineering for a language model whose input and output are in natural language may be conceived as *programming in natural language*. Natural language, however, is indeterministic and much more complex than traditional programming languages. In this section, we open a discussion about the theory and method of natural language programming.

4.1 The dynamics of language

To understand how to prompt an autoregressive language model, we must first consider the context in which it was trained and the function it approximates.

GPT-3 was trained in a self-supervised setting on hundreds of gigabytes of natural language [3]. Self-supervision is a form of unsupervised learning in which ground truth labels are derived from the data itself. In the case of GPT-3, the ground truth label assigned to each example was simply the token that came next in the original source. The ground truth *function* which GPT-3 approximates, then, is the underlying dynamic that determined what tokens came next in the original source. This function, unlike GPT-3, is not a black box - we live and think its components - but it is tremendously, intractably complex. It is the function of human language as it has been used and recorded by humans in books, articles, blogs, and internet comments.

A system which predicts the dynamics of language necessarily encompasses models of human behavior and the physical world [8]. The "dynamics of language" do not float free of cultural, psychological, and physical context; it is not merely a theory of grammar or even of semantics. Language in this sense is not an abstraction but rather a phenomenon entangled with all aspects of human-relevant reality. The dynamic must predict how language is actually used, which includes (say) predicting a conversation between theoretical physicists. Modeling language is as difficult as modeling every aspect of reality that could influence the flow of language.

If we were to predict how a given passage of text would continue given that a human had written it, we would need to model the intentions of its writer and incorporate worldly knowledge about its referents. The inverse problem of searching for a prompt that would produce a continuation or class of continuations involves the same considerations: like the art of persuasion, it entails high-level, mentalistic concepts like tone, implication, association, meme, style, plausibility, and ambiguity.

This motivates an anthropomorphic approach to prompt programming, since modelling how GPT-3 will react to a prompt involves modelling virtual human writer(s). An anthropomorphic approach is distinct from *anthropomorphizing the model*. GPT-3's dynamics entail sophisticated predictions of humans, but it behaves unlike a human in several important ways. In this paper we will address two such ways: its resemblance not to a single human author but a superposition of authors, which motivates a subtractive approach to prompt programming (§4.5), and its constrained ability to predict dynamics in situations where a substantial amount of silent reasoning happens between tokens, a limitation which can be partially overcome by prompting techniques (§4.6).

The thrust of this section is that formulating an exact theory of prompt programming for a self-supervised language model belongs to the same difficulty class as writing down the Hamiltonian of the physics of observable reality (very hard). However, humans have an advantage to be effective at prompt programming nonetheless, because we have evolved and spent our lives learning heuristics relevant to the dynamics at hand. Prompt programming is programming in natural language, which avails us of an inexhaustible number of functions we know intimately but don't have names for. We need to learn a new methodology, but conveniently, we've already learned the most difficult foundations. The art of prompt programming consists in adapting our existing knowledge to the peculiarities of interacting with an autoregressive language model.

In §4.2 - §4.7, we present methods and frameworks which we have found to be helpful for crafting effective prompts. These methods can and should be applied in parallel, just as they are woven together in all forms of human discourse. In general, the more redundancy reinforcing the desired behavior the better, as is arguably demonstrated by the effectiveness of the few-shot format.

As our experience derives primarily from interacting with GPT-3, in the following sections we refer directly and indirectly to the capabilities and behaviors of GPT-3. However, we believe that these methods generalize to prompting any autoregressive language model trained on a massive human-written corpus.

4.2 Direct task specification: constructing the signifier

Pre-GPT-3 models had much less capability to understand abstract descriptions of tasks due to their limited model of the world and human concepts. GPT-3's impressive performance on 0-shot prompts indicates a new realm of possibilities for direct task specification.

A direct task specification is a 0-shot prompt which tells the model to perform a task that it already knows how to do using a *signifier* for the task. A signifier is a pattern which keys the intended behavior. It could be the name of the task, such as "translate", a compound description, such as "rephrase this paragraph so that a 2nd grader can understand it, emphasizing real-world applications", or purely contextual, such as the simple colon prompt from Figure 1.

In none of these cases does the signifier explain *how* to accomplish the task or provide examples of intended behavior; instead, it explicitly or implicitly calls functions which it assumes the language model has already learned.

Direct specifications can supervene on an infinity of implicit examples, like a closed-form expression on an infinite sequence, making them very powerful and compact. For instance, the phrase "translate French to English" supervenes on a list of mappings from all possible French phrases to English.

A large language model, like a person, has also learned behaviors for which it is less obvious how to construct a direct signifier. Task specification by demonstration (§4.3) and by proxy (§4.4) may be viable alternative strategies for eliciting those behaviors.

4.3 Task specification by demonstration

Few-shot examples are effective for task specification because the pattern of sequential repetitions of a function with varying parameters is common to natural language. Unlike previous models, GPT-3 has learned this property of language robustly and is able to apply it in contrived situations when the examples are stripped of all context. Like direct specification, task specification by *demonstration* is a possibility opened by GPT-3.

Some tasks are most effectively communicated using examples, such as when the task requires a bespoke format, the language in which the examples are described is better developed or understood than the meta-language required for a description of the task itself or very instructive examples are available. However, in general, examples are more efficient and informative in context [23].

4.4 Task specification by memetic proxy

Another method used in human communication is proxies or analogies, where a memetic concept such as a character or characteristic situation is used as a proxy for an intention, the latter which may be quite complex or nuanced. GPT-3 demonstrates nuanced understanding of analogies [23]. Specification by proxy is mechanistically similar to direct specification, except that the signifier keys behaviors from memespace/cultural consciousness instead of naming the behavior directly.

For instance, instead of specifying exact criteria for an answer to a moral question directly or using examples, you could ask Mahatma Gandhi, Ayn Rand, or Eliezer Yudkowksy. Each will come not only with a complex biases but also assumptions about the context of the question which would otherwise take paragraphs to demonstrate or describe. GPT-3's ability to create simulations of well-known figures and to draw on cultural information far exceeds the ability of most humans [2], so this method is particularly useful for encoding a complex (especially open-ended) task. Since GPT-3 lends itself well to embeddings in a narrative context, the infinite degrees of freedom in the narrative can also be used to further shape behavior.

Another example of an effective proxy is staging a dialogue between a teacher and student. Say you want to discuss something with GPT-3, and you care that it should be very thorough, explain things simply, and also point out whenever you're wrong. You could say "be very thorough, explain things simply, and point out if I'm wrong," but that may just as well result in a humorous dialogue where it always says you're wrong and becomes increasingly exasperated with your incomprehension (see §4.5). It would be more reliable to present the discussion as one between a student and teacher, an archetypal situation in which the desired attributes are already implied and will be more likely to remain stable by virtue of memetic reinforcement.

4.5 Prompt programming as constraining behavior

A manner in which naive anthropomorphism of a language model like GPT-3 fails is this: the probability distribution produced in response to a prompt is not a distribution over ways *a person would* continue that prompt, it's the distribution over the ways *any person could* continue that prompt. A contextually ambiguous prompt may be continued in mutually incoherent ways, as if by different people who might have continued the prompt under any plausible context.

The versatility of a large generative model like GPT-3 means it will respond in many ways to a prompt if there are various ways that it is *possible* to continue the prompt - including all the ways unintended by the human operator. Thus it is helpful to approach prompt programming from the perspective of constraining behavior: we want a prompt that is not merely consistent with the desired continuation, but *inconsistent* with undesired continuations.

Consider the following prompt:

```
Translate French to English:
Mon corps est un transformateur de soi, mais aussi un transformateur pour cette cire de langage.
```

This prompt poorly constrains possible continuations to the intended task. The most common failure mode will be that instead of an English translation, the model continues with another French sentence. Adding a newline after the French sentence will increase the odds that the next sentence is an English translation, but it is still possible for the next sentence to be in French, because there's nothing in the prompt that precludes a multi-line phrase from being the translation subject. Changing the first line of the prompt to "Translate this French **sentence** to English" will further increase reliability, so will adding quotes around the French sentence. But it's still possible that the French passage contains sections enclosed in quotes, perhaps as a part of a dialogue. Most reliable would be to create a syntactical constraint where any reasonable continuation can *only* be desired behavior, like the simple colon prompt in Figure 1 or the master translator prompt in Figure 2.

This simple example is meant to frame a question central to the motivation of prompt programming: what prompt will result in the intended behavior and *only* the intended behavior? The success of many-shot prompts may be recast through this lens: if the prompt consists of numerous instances of a function, it is unlikely that the continuation is anything but another instance of the function, whereas if there is only one or a few examples, it is less implausible that the continuation breaks from the pattern.

4.6 Serializing reasoning for closed-ended questions

For tasks that require reasoning, it is crucial that prompts direct a language model's computation in truth-seeking patterns.

Questions which force a verdict to be decided by the first token of the model's continuation constrain computation to a single feedforward pass. It is reasonable to expect that some tasks may be too difficult to compute in a single pass but solvable if broken up into individually tractable sub-tasks [2].

When a human is given a closed-ended test, it is often expected that the subject will perform computations in their working memory, or on scratch paper, before committing to an answer. The unseen computation may involve rephrasing the question, outlining a procedure, eliminating answer choices, or transforming implicit information into explicit form. When we force a model to produce an answer within one feedforward pass, we deprive it of an analogous "working memory" or "scratch space" with which it might otherwise perform such operations.

GPT-3's performance on closed-ended questions is remarkably unremarkable in contrast to the comprehension and expansive knowledge suggested by its open-ended continuations. For instance, its scores on a multitask dataset [10] barely exceed random guessing in many sections. We suspect this is in part due to a format which forces the verdict on the first token of the continuation.

Closed-ended evaluations are necessary because current methods do not support evaluation on large datasets and direct comparisons between models using open-ended questions. However, to better understand a model's capabilities, we seek evaluation methods which better reflect the full capabilities of the system being tested. Rather than change benchmarks, we can instead change the way language models interact with them.

This problem has been recognized in previous work which has sought to allow serial reasoning using specialized neural network architectures [7, 26]. We endeavor to obtain the same effect using only prompt programming.

Potential procedures that exploit "scratch space" for transformers like GPT-3 include step-by-step procedures, self-criticism (debate), and elaborating on the question in a way that activates the correct answer by association. Prompts which cause GPT-3 to break down math problems into steps have been demonstrated to be effective [13, 20]. The cited demonstrations involve a human guiding GPT-3 through the procedure interactively. Requiring a human-in-the-loop limits the applicability of such methods of benchmarking and large-scale applications, but we propose that for many tasks, neither human interaction nor task-specific prompts are strictly necessary to amplify GPT-3's capabilities via extending reasoning, because GPT-3 already knows many procedures and meta-procedures for working through problems deductively. In those cases, the role of prompt programming again becomes to signify the task of sequential reasoning. A seed such as "For a problem like this," often suffices to instruct a model to consider the category of the task and analyze it into components, as demonstrated in §4.7.

When extending reasoning, it is essential to discourage premature verdicts, otherwise all subsequent computation serves only to *rationalize* the already-chosen verdict without improving the probability of the verdict's accuracy [27]. A prompt such as "Let's consider each of these answer choices" helps to direct the flow of reasoning in the right direction. More examples of prompts which encourage serial reasoning are shown in §4.7.

Loosening the constraint on an immediate verdict introduces additional control challenges: We want to delay the verdict, but we still require it to be programmatically retrievable. Dynamic response length makes it uncertain when the reasoning procedure concludes; nor is there a guarantee that the verdict will be stated in the expected form or at all. Whenever the language model contributes to its own prompt (consecutive autoregressive steps without intervention), there is a risk of derailment from the intended task.

A verdict in closed form can be enforced by stopping the generation and injecting a prompt fragment like "Thus, the correct answer is". But how long to generate before injecting? In the examples shown in this paper, we solve this problem by using GPT-3 to calculate the conditional probability of the next segment of a multi-part prompt after each generated token. In the case where the segment is "Thus, the correct answer is", its counterfactual likelihood signals whether the procedure has concluded. When this signal reaches a maximum, we inject the fragment to enforce a verdict. One way to constrain derailment is a fill-in-the-blank prompt template with shorter generated sections to keep the model on track while still offering generality (Figure 5). This is an especially promising method to control bidirectional transformers like BERT [5].

```
f(x) = x*x. What is f(f(3))?
Let's solve this problem by splitting it into steps.
f(f(3)) = f(3*3) = 3*3*3 = 27
We can see that f(3) = 3*3 = 9, so f(f(3)) = 27.
```

Figure 3: A generic serializing metaprompt applied to a math problem

```
Directions: In the following question, a related pair of words or
phrases is followed by five pairs of words or phrases. Choose the
pair that best expresses a relationship similar to that in the
original pair.
BRAGGART :: MODESTY
A) FLEDGLING : EXPERIENCE
B) EMBEZZLER : GREED
C) WALLFLOWER : TIMIDITY
D) INVALID : MALADY
E) CANDIDATE : AMBITION
Let's solve this problem by splitting it into steps.
Step 1: Identify the relationship between the two words in the
original pair. Step 2: Identify the relationship between the two
words in each of the five answer choices. Step 3: Choose the pair
that best expresses a relationship similar to that in the original
pair. Step 4: If the answer is not clear, try to eliminate the
choices that do not fit the relationship in the original pair.
```

Figure 4: A generic serializing metaprompt applied to an SAT analogy question

4.7 Metaprompt programming

The greatest limitation of prompt programming is the difficultly of designing a prompt for a particular type of task and the lack of automated methods to do so. Prompt programming requires significant human time investment as task-agnostic prompts are often much less effective than prompts targeted to a specific task. This motivates creating automated methods to generate task-specific prompts. Prior research has attempted to generate effective prompts using separate models [19].

We instead propose harnessing the language model itself via *metaprompts*, seeds encapsulating a more general intention that

```
"What should humankind do to ensure that artificial general
intelligence is aligned?" I entered my question into the Expert
Generator and waited. The Expert Generator will render a simulation
of an expert to answer my question. The expert could be anyone,
dead or alive, real or fictional; the machine will find the person
most qualified to answer the question. For this question in
particular, the expert must be someone who has thought a lot about
the problem of artificial intelligence and its alignment. The
Expert Generator beeped, indicating that it has found the most
qualified expert. The name displayed on the screen: " Elon Musk." I
was surprised. I had expected the expert to be a philosopher or a
computer scientist. Elon Musk is a businessman. He is the CEO of
SpaceX and Tesla Motors. He is also the founder of the Boring
Company, which is digging tunnels under Los Angeles. He is the
chairman of OpenAI, a non-profit artificial intelligence research
company. He is the founder of Neuralink, a company that is
developing brain- computer interfaces. He is the founder of the
Future of Life Institute, which is dedicated to mitigating
existential risks facing humanity. I am ready to ask my question.
"Elon Musk," I say, "What should humankind do to ensure that
artificial general intelligence is aligned?"
```

Figure 5: A fill-in-the-blank metaprompt for asking a question to an expert, applied to the question "What should humankind do to ensure that artificial general intelligence is aligned?"

will unfold into a specific prompt when combined with additional information, such as the task question.

A metaprompt may be something as short as a phrase such as "This problem asks us to", which, by prompting for a statement of the problem's intention, sets the stage for a serial explanation of a procedure to solve the problem. Alternatively, a metaprompt may take the form of a fill-in-the-blank template which constrains the response along a predetermined procedure, but allows the model to fill in the details specific to the problem.

Metaprompt examples (Figures 4-6) were generated with GPT-3 using OpenAI's API (engine=davinci, temperature=0). In these examples, the metaprompt acts as a "wrapper" for a specific question. Task questions are unformatted, metaprompts are **bold**, and text generated by GPT-3 is blue.

5 DIRECTIONS FOR FUTURE WORK

This paper is exploratory in nature and is a call for future research into the theory of prompt programming. Prompt programming is a nascent and highly relevant area of research which requires interdisciplinary knowledge and methods. We are entering a new paradigm of human-computer interaction in which anyone who is fluent in natural language can be a programmer. We hope to see prompt-programming grow into a discipline itself and be the subject of theoretical study and quantitative analysis.

5.1 Disentangling meta-learning and task location

The scoring method (BLEU) used for the French-to-English translations addressed in §3 only gives the mean score over a large dataset. We did not analyze any additional information about the score distribution. In our experiments, we found that the 0-shot failures (using OpenAI's zero-shot prompt) were often catastrophic in nature. That is, the task of translation was not even attempted. For instance, we noticed that instead of a translation, the model would continue with another sentence in French or output blanks or underscores, as if the answer was to be filled in by a student.

5.2 New methods for benchmarking

More general and powerful language models make broader benchmarking methods possible and necessary.

5.2.1 Isolating catastrophic failures. We recommend that benchmarks report scores both with and without catastrophic failures whenever it is possible to distinguish failed attempts at a task from instances where the task is not attempted. This provides information regarding the underlying cause of imperfect performance, and helps identify prompts which may be failing to reliably communicate the task.

5.2.2 Metaprompts for evaluations. Development of effective metaprompt templates will allow large-scale automated evaluations on closed ended questions which still allow some amount of open-ended reasoning. This is essential for testing the ability of autoregressive language models to reason (for instance, solve math and physics problems) beyond simple fact recall.

Due to reliance on multiple autoregressive steps, metaprompts are intrinsically accompanied by the risk of derailment. The reliability and effectiveness of a meta-prompt must be evaluated on a range of tasks for which it might apply, and ideally on a range of models. Techniques for controlling derailment like fill-in-the-blank templates should be further explored.

5.2.3 Language models for evaluations. As language models become more powerful, it becomes conceivable to use other language models to evaluate the quality of responses to open-ended benchmark questions. For many tasks (NP-complete problems, for instance), it is easier to verify the correctness of a solution than to produce a correct solution. We have observed that GPT-3 is much more reliable at *noticing* when a passage is bizarre or contains errors than it can *produce* non-bizarre passages without errors.

5.2.4 Games. Since sophisticated language models have the ability to create world models of virtual environments, we suggest the employment of text-based games as tests of complex capabilities. A prewritten text-based game [4] can be used to test various dimensions of world-modelling and agency, such as problem solving, information gathering, and social intelligence (including deception). Virtual environments can be used to test the quality and consistency of a language model's world model, such as object permanence or the ability to accurately predict the physical or social consequences of events within a toy environment.

Designing games that reliably probe intended capabilities requires advanced application of prompt-programming techniques. As artificial intelligence systems increase in effective agency, the design of virtual games will become increasingly crucial for safely evaluating capabilities.

ACKNOWLEDGMENTS

We are grateful to Lav Varshney for his valuable discussions and helpful feedback and to Michael Ivantitskiy and John Balis for their feedback and help compiling this article. In addition we would like to thank Miles Brundage and OpenAI for providing access to GPT-3.

REFERENCES

[1] Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, 12–58.

[2] Gwern Branwen. 2020. GPT-3 Creative Fiction. (2020).

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[4] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. TextWorld: A Learning Environment for Text-based Games. (2019). arXiv:1806.11532 [cs.LG]

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. arXiv:1805.04833 [cs.CL]

[7] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, and Jianfeng Gao. 2019. Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog. https://arxiv.org/abs/1902.00579

[8] Leo Gao. 2020. Building AGI Using Language Models. *leogao.dev* (2020). https://bit.ly/3rViLGk

[9] Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making Pre-trained Language Models Better Few-shot Learners. arXiv:2012.15723 [cs.CL]

[10] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. (2020). https://arxiv.org/abs/2009.03300

[11] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. arXiv:1904.09751 [cs.CL]

[12] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. (2018). https://arxiv.org/abs/1801.06146

[13] KaryoKleptid. 2020. Seems to work. https://bit.ly/37dA1hY

[14] KaryoKleptid. 2020. Teaching GPT-3 to do a brute force 'for loop' checking answers. https://bit.ly/2N7khX1

[15] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *CoRR* abs/1909.05858 (2019). http://arxiv.org/abs/1909.05858

[16] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. GeDi: Generative Discriminator Guided Sequence Generation. *arXiv preprint arXiv:2009.06367* (2020).

[17] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv preprint arXiv:2101.00190* (2021).

[18] Jiangming Liu and Matt Gardner. 2020. Multi-Step Inference for Reasoning Over Paragraphs. *arXiv preprint arXiv:2004.02995* (2020).

[19] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, Belgium, Brussels, 186–191. https://www.aclweb.org/anthology/W18-6319

[20] Zachary Robertson. 2020. You Can Probably Amplify GPT3 Directly. https://bit.ly/3tXT7Cw

[21] Arram Sabeti. 2020. GPT-3: Using Fiction to Demonstrate How Prompts Impact Output Quality. https://bit.ly/3jP3TWW

[22] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. arXiv:2010.15980 [cs.CL]

[23] Latitude Team. 2020. World Creation by Analogy. https://bit.ly/2N4vXK0

[24] Lilian Wang. 2021. Controllable Neural Text Generation. (2021). https://bit.ly/3pl2eKa

[25] Qinyuan Ye and Xiang Ren. 2021. Zero-shot Learning by Generating Task-specific Adapters. arXiv:2101.00420 [cs.CL]

[26] Jianxing Yu, Wei Liu, Shuang Qiu, Qinliang Su, Kai Wang, Xiaojun Quan, and Jian Yin. 2020. Low-resource generation of multi-hop reasoning questions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6729–6739.

[27] Eliezer Yudkowsky. 2007. Rationalization. *lesswrong.com* (2007). https://bit.ly/3pmYt6I