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

Laria Reynolds
moire@knc.ai

Kyle McDonell
kyle@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. In this work, 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.

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

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 a small number of examples of solved tasks are provided as part of the input to the trained model. However, while the few-shot format was sufficient to reveal surprising performance on these tasks, we argue that prompting can be more effective than either fine-tuning or the few-shot format at extracting specific learned behaviors from self-supervised language models.

We argue that contrary to the common interpretation of the few-shot format implied by the title of the original GPT-3 paper [3], *Language models are few-shot learners*, 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 or instruction, 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 an self-supervised model in the modality it was trained: natural language.

The ground truth function that self-supervised language models are trained to approximate is, in great generality, is how humans write. Accordingly, to interact with and control a language model, we should consider doing so from the perspective of natural language as it is used by humans. 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 paper, 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 optimization of prompts [22, 17], 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 [18, 9].

In comparison, little work on natural-language, 0-shot 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 a brief, non-exhaustive survey of explorations which have gone beyond the few-shot paradigm.

Gwern has given the most comprehensive survey of GPT-3's capabilities through demonstrations of it writing fiction, poetry, navy seals copypasta parodies, 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]. Arram Sabeti has written about the effect of the context provided by a prompt on writing quality [21]. Zachary 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 and this allows it to perform better than with fewer or no examples [3].

The improvement in performance with the number of examples, however, can be interpreted in an alternate way. Rather than learning how to perform the task from the examples, the examples may simply serve to instruct GPT-3 on what task it is to solve and encourage it to follow the structure of the prompt in its response.

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 and time 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 in the original GPT-3 paper [3] can be improved substantially with even minor prompt engineering. Most significantly, the extremely simple prompt in Figure (1) which includes only the names of the two languages and a colon performs better than the 10-shot prompt in the style of the original GPT-3 paper.

In fact, we found this pattern was true of most of the worst-performing 0-shot prompts in the original GPT-3 paper [3], particularly question and answer benchmarks. Many could easily be improved by simple changes in formatting which make the prompt closer to natural language as a human would write it. Thus, GPT-3's 0-shot or baseline performance without meta-learning was significantly underestimated.

It is important to correct this confusion to get a more precise understanding of the nature of a model's capabilities so that we can better learn to control it. The fact that GPT-3 has a vast repertoire of functions that do not need to be learned at runtime allows for great flexibility in 0-shot 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) 1-shot performed significantly worse than 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 the example as a categorical guide, it is inferred that the semantic meaning of the examples are relevant to the task, e.g. the example is 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].

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

Table 1: We report BLEU scores for variants of the GPT-3 model using different prompt formats on the WMT'14 Fr-En dataset [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 or because of unknown hyperparameters. Additionally, the size of the *Babbage* and *Curie* models are not reported so the relationship to the models in the original GPT-3 paper is inferred. We were unable to replicate the 64-shot test due to API constraints and instead replaced it with a 10-shot test.

```
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.

```
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.

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.

GPT-3 has not learned the ground truth function perfectly, obviously, or else the world would look very different by now. However, it has approximated it to a notable extent, as evidenced by its ability to not only form grammatical sentences, but also coherently employ cultural references and metaphors and model complex psychological and physical contexts [2]. The problem of prompt programming, then, is nontrivial, for the dynamics of language (or an approximation thereof on GPT-3's level of sophistication) are nontrivial.

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 some task that it already knows how to do. A direct specification consists in constructing 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.

It is important to note that unlike in fine-tuning, the "training examples" in few-shot are processed as a whole, and may not necessarily be interpreted as parallel and independent. Informative context or a large number of examples can help mitigate the problems with few-shot addressed in §3.2. For instance, a prompt could embed examples in a context which makes it clear that the examples are independent instances of a function rather than a sequential pattern that should be extrapolated. In general, examples are more efficient and informative in context, both from the perspective of a human and a language model [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 may be 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 does poorly at constraining 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 of all would be to create a syntactical constraint where any reasonable continuation can only be desired behavior, like the simple colon prompt from Figure 1 and the master translator prompt from 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 feed-forward 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 robust comprehension and expansive knowledge suggested by its open-ended continuations. For instance, its scores on this multitask dataset [10] barely exceed random guessing for some 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 [26, 7]. 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 [20, 13]. 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 in a programmatically retrievable form. 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 6). This is an especially promising method to control bidirectional transformers like BERT [5].

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 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", a seemingly innocuous fragment 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 (Figs 3-5) 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.

```
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

```
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

In order to solve this problem, we will analyze
each of the options and determine which one is the
best fit. Let's begin.
A) FLEDGLING : EXPERIENCE

Fledgling is a young bird that has just learned to fly.
Experience is the knowledge gained from one's life.
```

Figure 5: A fill-in-the-blank serializing metaprompt for multiple-choice questions applied to an SAT analogy question. The response alternates between generated text (unformatted) and prompt (bold).

```
"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 6: 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?"

5 Directions for future work

This paper is exploratory in nature and is a call for future research into the theory of prompt programming and creation of automated methods of prompting.

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.

The hypothesis that the examples are performing task location suggests that if the catastrophic failures were removed from the score, performance on 0 and 64-shot prompts will become more similar, if not equivalent. Furthermore, we suspect that performance on 1-shot prompts will be significantly worse than on 0 and 64-shot prompts due to the phenomena of content leakage and faulty generalization addressed in §3.2.

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 meta-prompt 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, for instance, 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 safety evaluating capabilities.

Acknowledgements

We are grateful to Lav Varshney for his valuable discussions and helpful feedback and to Michael Ivanitskiy 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 et al. "Findings of the 2014 Workshop on Statistical Machine Translation". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland, USA: Association for Computational Linguistics, June 2014, pp. 12–58.

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

[3] Tom B Brown et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[4] Marc-Alexandre Côté et al. "TextWorld: A Learning Environment for Text-based Games". In: (2019). arXiv: 1806.11532 [cs.LG].

[5] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

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

[7] Zhe Gan et al. *Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog*. 2019. arXiv: 1902.00579 [cs.CV]. URL: https://arxiv.org/abs/1902.00579.

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

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

[10] Dan Hendrycks et al. "Measuring massive multitask language understanding". In: *arXiv preprint arXiv:2009.03300* (2020). URL: https://arxiv.org/abs/2009.03300.

[11] Ari Holtzman et al. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL].

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

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

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

[15] Nitish Shirish Keskar et al. "CTRL: A Conditional Transformer Language Model for Controllable Generation". In: *CoRR* abs/1909.05858 (2019). arXiv: 1909.05858. URL: http://arxiv.org/abs/1909.05858.

[16] Ben Krause et al. "GeDi: Generative Discriminator Guided Sequence Generation". In: *arXiv preprint arXiv:2009.06367* (2020).

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

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

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

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

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

[22] Taylor Shin et al. *AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts*. 2020. arXiv: 2010.15980 [cs.CL].

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

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

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

[26] Jianxing Yu et al. "Low-resource generation of multi-hop reasoning questions". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 6729–6739.

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

Language Models are Few-Shot Learners

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** **Dario Amodei**

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

*Equal contribution

†Johns Hopkins University, OpenAI

Author contributions listed at end of paper.

Contents

1 Introduction

Recent years have featured a trend towards pre-trained language representations in NLP systems, applied in increasingly flexible and task-agnostic ways for downstream transfer. First, single-layer representations were learned using word vectors [MCCD13, PSM14] and fed to task-specific architectures, then RNNs with multiple layers of representations and contextual state were used to form stronger representations [DL15, MBXS17, PNZtY18] (though still applied to task-specific architectures), and more recently pre-trained recurrent or transformer language models [VSP+17] have been directly fine-tuned, entirely removing the need for task-specific architectures [RNSS18, DCLT18, HR18].

This last paradigm has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, and many others, and has continued to advance based on new architectures and algorithms [RSR+19, LOG+19, YDY+19, LCG+19]. However, a major limitation to this approach is that while the architecture is task-agnostic, there is still a need for task-specific datasets and task-specific fine-tuning: to achieve strong performance on a desired task typically requires fine-tuning on a dataset of thousands to hundreds of thousands of examples specific to that task. Removing this limitation would be desirable, for several reasons.

First, from a practical perspective, the need for a large dataset of labeled examples for every new task limits the applicability of language models. There exists a very wide range of possible useful language tasks, encompassing anything from correcting grammar, to generating examples of an abstract concept, to critiquing a short story. For many of these tasks it is difficult to collect a large supervised training dataset, especially when the process must be repeated for every new task.

Second, the potential to exploit spurious correlations in training data fundamentally grows with the expressiveness of the model and the narrowness of the training distribution. This can create problems for the pre-training plus fine-tuning paradigm, where models are designed to be large to absorb information during pre-training, but are then fine-tuned on very narrow task distributions. For instance [HLW+20] observe that larger models do not necessarily generalize better out-of-distribution. There is evidence that suggests that the generalization achieved under this paradigm can be poor because the model is overly specific to the training distribution and does not generalize well outside it [YdC+19, MPL19]. Thus, the performance of fine-tuned models on specific benchmarks, even when it is nominally at human-level, may exaggerate actual performance on the underlying task [GSL+18, NK19].

Third, humans do not require large supervised datasets to learn most language tasks – a brief directive in natural language (e.g. "please tell me if this sentence describes something happy or something sad") or at most a tiny number of demonstrations (e.g. "here are two examples of people acting brave; please give a third example of bravery") is often

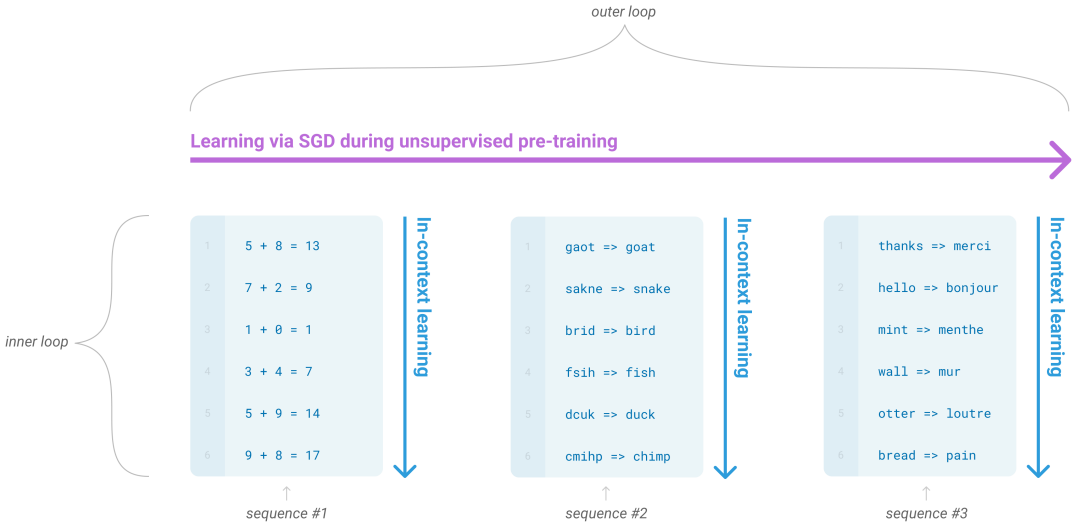


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

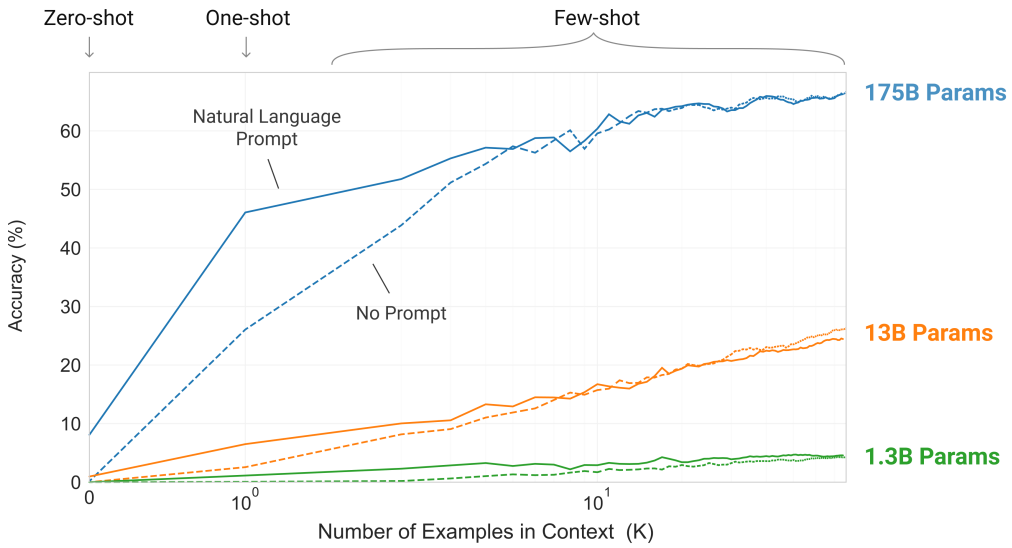


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

sufficient to enable a human to perform a new task to at least a reasonable degree of competence. Aside from pointing to a conceptual limitation in our current NLP techniques, this adaptability has practical advantages – it allows humans to seamlessly mix together or switch between many tasks and skills, for example performing addition during a lengthy dialogue. To be broadly useful, we would someday like our NLP systems to have this same fluidity and generality.

One potential route towards addressing these issues is meta-learning[1] – which in the context of language models means the model develops a broad set of skills and pattern recognition abilities at training time, and then uses those abilities at inference time to rapidly adapt to or recognize the desired task (illustrated in Figure 1.1). 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.

While it has shown some initial promise, this approach still achieves results far inferior to fine-tuning – for example [RWC+19] achieves only 4% on Natural Questions, and even its 55 F1 CoQa result is now more than 35 points behind the state of the art. Meta-learning clearly requires substantial improvement in order to be viable as a practical method of solving language tasks.

Another recent trend in language modeling may offer a way forward. In recent years the capacity of transformer language models has increased substantially, from 100 million parameters [RNSS18], to 300 million parameters [DCLT18], to 1.5 billion parameters [RWC+19], to 8 billion parameters [SPP+19], 11 billion parameters [RSR+19], and finally 17 billion parameters [Tur20]. Each increase has brought improvements in text synthesis and/or downstream NLP tasks, and there is evidence suggesting that log loss, which correlates well with many downstream tasks, follows a smooth trend of improvement with scale [KMH+20]. Since in-context learning involves absorbing many skills and tasks within the parameters of the model, it is plausible that in-context learning abilities might show similarly strong gains with scale.

[1] In the context of language models this has sometimes been called "zero-shot transfer", but this term is potentially ambiguous: the method is "zero-shot" in the sense that no gradient updates are performed, but it often involves providing inference-time demonstrations to the model, so is not truly learning from zero examples. 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.

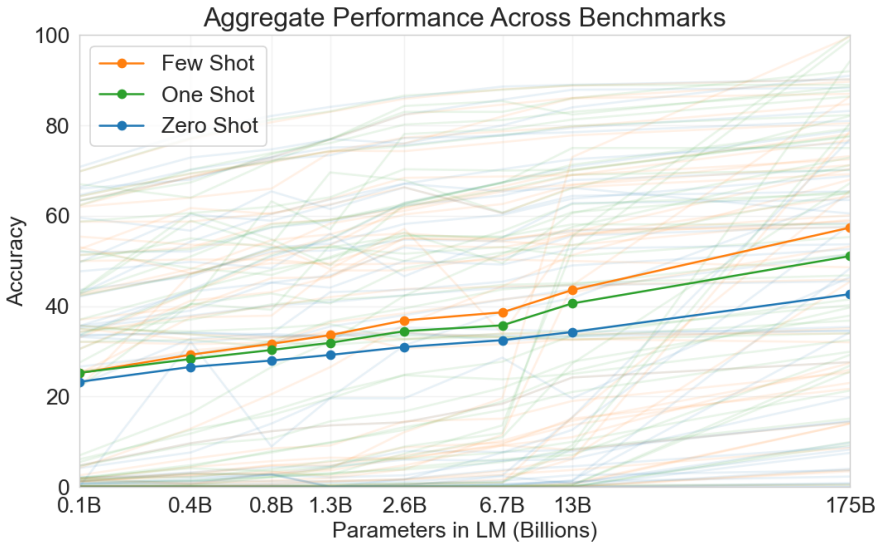


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

In this paper, we test this hypothesis by training a 175 billion parameter autoregressive language model, which we call GPT-3, and measuring its in-context learning abilities. Specifically, we evaluate GPT-3 on over two dozen NLP datasets, as well as several novel tasks designed to test rapid adaptation to tasks unlikely to be directly contained in the training set. 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. GPT-3 could also in principle be evaluated in the traditional fine-tuning setting, but we leave this to future work.

Figure 1.2 illustrates the conditions we study, and shows few-shot learning of a simple task requiring the model to remove extraneous symbols from a word. Model performance improves with the addition of a natural language task description, and with the number of examples in the model's context, K. Few-shot learning also improves dramatically with model size. Though the results in this case are particularly striking, the general trends with both model size and number of examples in-context hold for most tasks we study. We emphasize that these "learning" curves involve no gradient updates or fine-tuning, just increasing numbers of demonstrations given as conditioning.

Broadly, on NLP tasks GPT-3 achieves promising results in the zero-shot and one-shot settings, and in the the few-shot setting is sometimes competitive with or even occasionally surpasses state-of-the-art (despite state-of-the-art being held by fine-tuned models). For example, GPT-3 achieves 81.5 F1 on CoQA in the zero-shot setting, 84.0 F1 on CoQA in the one-shot setting, 85.0 F1 in the few-shot setting. Similarly, GPT-3 achieves 64.3% accuracy on TriviaQA in the zero-shot setting, 68.0% in the one-shot setting, and 71.2% in the few-shot setting, the last of which is state-of-the-art relative to fine-tuned models operating in the same closed-book setting.

GPT-3 also displays one-shot and few-shot proficiency at tasks designed to test rapid adaption or on-the-fly reasoning, which include unscrambling words, performing arithmetic, and using novel words in a sentence after seeing them defined only once. We also show that in the few-shot setting, GPT-3 can generate synthetic news articles which human evaluators have difficulty distinguishing from human-generated articles.

At the same time, we also find some tasks on which few-shot performance struggles, even at the scale of GPT-3. This includes natural language inference tasks like the ANLI dataset, and some reading comprehension datasets like RACE or QuAC. By presenting a broad characterization of GPT-3's strengths and weaknesses, including these limitations, we hope to stimulate study of few-shot learning in language models and draw attention to where progress is most needed.

A heuristic sense of the overall results can be seen in Figure 1.3, which aggregates the various tasks (though it should not be seen as a rigorous or meaningful benchmark in itself).

We also undertake a systematic study of "data contamination" – a growing problem when training high capacity models on datasets such as Common Crawl, which can potentially include content from test datasets simply because such content often exists on the web. In this paper we develop systematic tools to measure data contamination and quantify its distorting effects. Although we find that data contamination has a minimal effect on GPT-3's performance on most datasets, we do identify a few datasets where it could be inflating results, and we either do not report results on these datasets or we note them with an asterisk, depending on the severity.

In addition to all the above, we also train a series of smaller models (ranging from 125 million parameters to 13 billion parameters) in order to compare their performance to GPT-3 in the zero, one and few-shot settings. Broadly, for most tasks we find relatively smooth scaling with model capacity in all three settings; one notable pattern is that the gap between zero-, one-, and few-shot performance often grows with model capacity, perhaps suggesting that larger models are more proficient meta-learners.

Finally, given the broad spectrum of capabilities displayed by GPT-3, we discuss concerns about bias, fairness, and broader societal impacts, and attempt a preliminary analysis of GPT-3's characteristics in this regard.

The remainder of this paper is organized as follows. In Section 2, we describe our approach and methods for training GPT-3 and evaluating it. Section 3 presents results on the full range of tasks in the zero-, one- and few-shot settings. Section 4 addresses questions of data contamination (train-test overlap). Section 5 discusses limitations of GPT-3. Section 6 discusses broader impacts. Section 7 reviews related work and Section 8 concludes.

2 Approach

Our basic pre-training approach, including model, data, and training, is similar to the process described in [RWC+19], with relatively straightforward scaling up of the model size, dataset size and diversity, and length of training. Our use of in-context learning is also similar to [RWC+19], but in this work we systematically explore different settings for learning within the context. Therefore, we start this section by explicitly defining and contrasting the different settings that we will be evaluating GPT-3 on or could in principle evaluate GPT-3 on. These settings can be seen as lying on a spectrum of how much task-specific data they tend to rely on. Specifically, we can identify at least four points on this spectrum (see Figure 2.1 for an illustration):

- **Fine-Tuning (FT)** has been the most common approach in recent years, and involves updating the weights of a pre-trained model by training on a supervised dataset specific to the desired task. Typically thousands to hundreds of thousands of labeled examples are used. The main advantage of fine-tuning is strong performance on many benchmarks. The main disadvantages are the need for a new large dataset for every task, the potential for poor generalization out-of-distribution [MPL19], and the potential to exploit spurious features of the training data [GSL+18, NK19], potentially resulting in an unfair comparison with human performance. In this work we do not fine-tune GPT-3 because our focus is on task-agnostic performance, but GPT-3 can be fine-tuned in principle and this is a promising direction for future work.

- **Few-Shot (FS)** is the term we will use in this work to refer to the setting where the model is given a few demonstrations of the task at inference time as conditioning [RWC+19], but no weight updates are allowed. As shown in Figure 2.1, for a typical dataset an example has a context and a desired completion (for example an English sentence and the French translation), and few-shot works by giving K examples of context and completion, and then one final example of context, with the model expected to provide the completion. We typically set K in the range of 10 to 100 as this is how many examples can fit in the model's context window ($n_{\text{ctx}} = 2048$). The main advantages of few-shot are a major reduction in the need for task-specific data and reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset. The main disadvantage is that results from this method have so far been much worse than state-of-the-art fine-tuned models. Also, a small amount of task specific data is still required. As indicated by the name, few-shot learning as described here for language models is related to few-shot learning as used in other contexts in ML [HYC01, VBL+16] – both involve learning based on a broad distribution of tasks (in this case implicit in the pre-training data) and then rapidly adapting to a new task.

- **One-Shot (1S)** is the same as few-shot except that only one demonstration is allowed, in addition to a natural language description of the task, as shown in Figure 1. The reason to distinguish one-shot from few-shot and zero-shot (below) is that it most closely matches the way in which some tasks are communicated to humans. For example, when asking humans to generate a dataset on a human worker service (for example Mechanical Turk), it is common to give one demonstration of the task. By contrast it is sometimes difficult to communicate the content or format of a task if no examples are given.

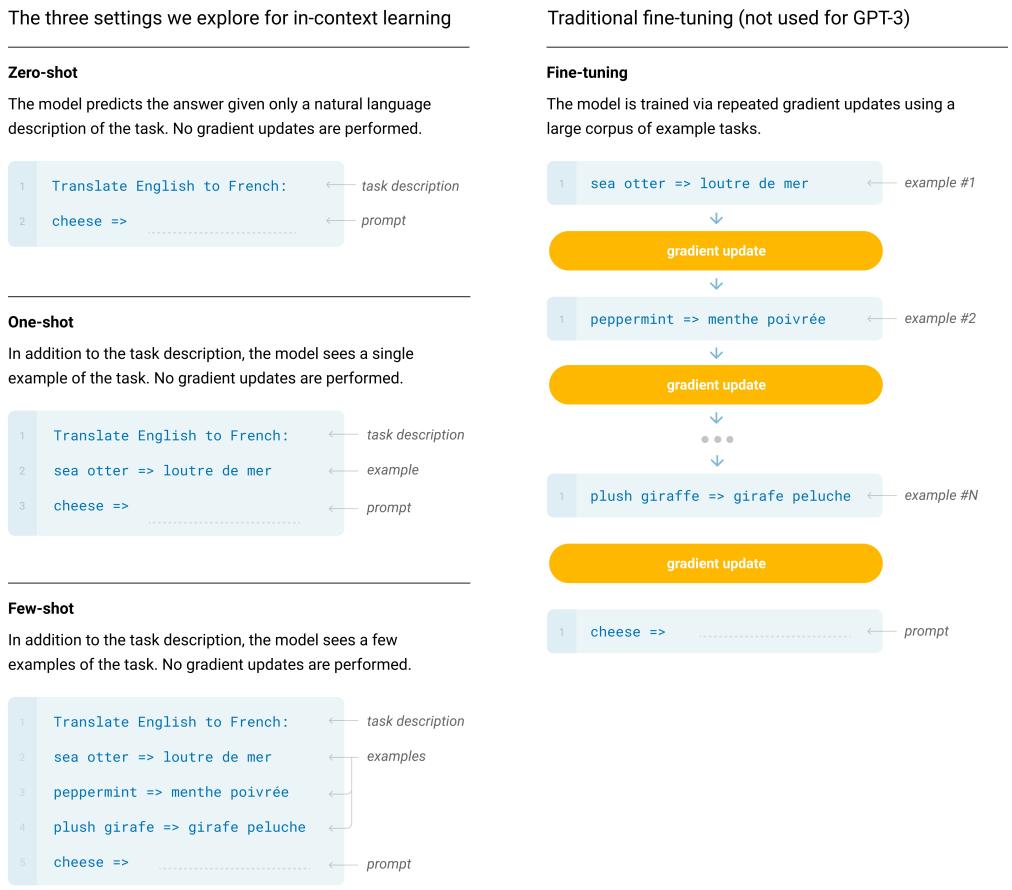


Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning. The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-, and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting. Exact phrasings for all task descriptions, examples and prompts can be found in Appendix G.

- **Zero-Shot (0S)** is the same as one-shot except that no demonstrations are allowed, and the model is only given a natural language instruction describing the task. This method provides maximum convenience, potential for robustness, and avoidance of spurious correlations (unless they occur very broadly across the large corpus of pre-training data), but is also the most challenging setting. In some cases it may even be difficult for humans to understand the format of the task without prior examples, so this setting is in some cases "unfairly hard". For example, if someone is asked to "make a table of world records for the 200m dash", this request can be ambiguous, as it may not be clear exactly what format the table should have or what should be included (and even with careful clarification, understanding precisely what is desired can be difficult). Nevertheless, for at least some settings zero-shot is closest to how humans perform tasks – for example, in the translation example in Figure 2.1, a human would likely know what to do from just the text instruction.

Figure 2.1 shows the four methods using the example of translating English to French. In this paper we focus on zero-shot, one-shot and few-shot, with the aim of comparing them not as competing alternatives, but as different problem settings which offer a varying trade-off between performance on specific benchmarks and sample efficiency. We especially highlight the few-shot results as many of them are only slightly behind state-of-the-art fine-tuned models. Ultimately, however, one-shot, or even sometimes zero-shot, seem like the fairest comparisons to human performance, and are important targets for future work.

Sections 2.1-2.3 below give details on our models, training data, and training process respectively. Section 2.4 discusses the details of how we do few-shot, one-shot, and zero-shot evaluations.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

2.1 Model and Architectures

We use the same model and architecture as GPT-2 [RWC+19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, ranging over three orders of magnitude from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3. Previous work [KMH+20] suggests that with enough training data, scaling of validation loss should be approximately a smooth power law as a function of size; training models of many different sizes allows us to test this hypothesis both for validation loss and for downstream language tasks.

Table 2.1 shows the sizes and architectures of our 8 models. Here n_{params} is the total number of trainable parameters, n_{layers} is the total number of layers, d_{model} is the number of units in each bottleneck layer (we always have the feedforward layer four times the size of the bottleneck layer, $d_{\text{ff}} = 4 * d_{\text{model}}$), and d_{head} is the dimension of each attention head. All models use a context window of $n_{\text{ctx}} = 2048$ tokens. We partition the model across GPUs along both the depth and width dimension in order to minimize data-transfer between nodes. The precise architectural parameters for each model are chosen based on computational efficiency and load-balancing in the layout of models across GPU's. Previous work [KMH+20] suggests that validation loss is not strongly sensitive to these parameters within a reasonably broad range.

2.2 Training Dataset

Datasets for language models have rapidly expanded, culminating in the Common Crawl dataset[2] [RSR+19] constituting nearly a trillion words. This size of dataset is sufficient to train our largest models without ever updating on the same sequence twice. However, we have found that unfiltered or lightly filtered versions of Common Crawl tend to have lower quality than more curated datasets. Therefore, we took 3 steps to improve the average quality of our datasets: (1) we downloaded and filtered a version of CommonCrawl based on similarity to a range of high-quality reference corpora, (2) we performed fuzzy deduplication at the document level, within and across datasets, to prevent redundancy and preserve the integrity of our held-out validation set as an accurate measure of overfitting, and (3) we also added known high-quality reference corpora to the training mix to augment CommonCrawl and increase its diversity.

Details of the first two points (processing of Common Crawl) are described in Appendix A. For the third, we added several curated high-quality datasets, including an expanded version of the WebText dataset [RWC+19], collected by scraping links over a longer period of time, and first described in [KMH+20], two internet-based books corpora (Books1 and Books2) and English-language Wikipedia.

Table 2.2 shows the final mixture of datasets that we used in training. The CommonCrawl data was downloaded from 41 shards of monthly CommonCrawl covering 2016 to 2019, constituting 45TB of compressed plaintext before filtering and 570GB after filtering, roughly equivalent to 400 billion byte-pair-encoded tokens. Note that during training, datasets are not sampled in proportion to their size, but rather datasets we view as higher-quality are sampled more frequently, such that CommonCrawl and Books2 datasets are sampled less than once during training, but the other datasets are sampled 2-3 times. This essentially accepts a small amount of overfitting in exchange for higher quality training data.

[2] https://commoncrawl.org/the-data/

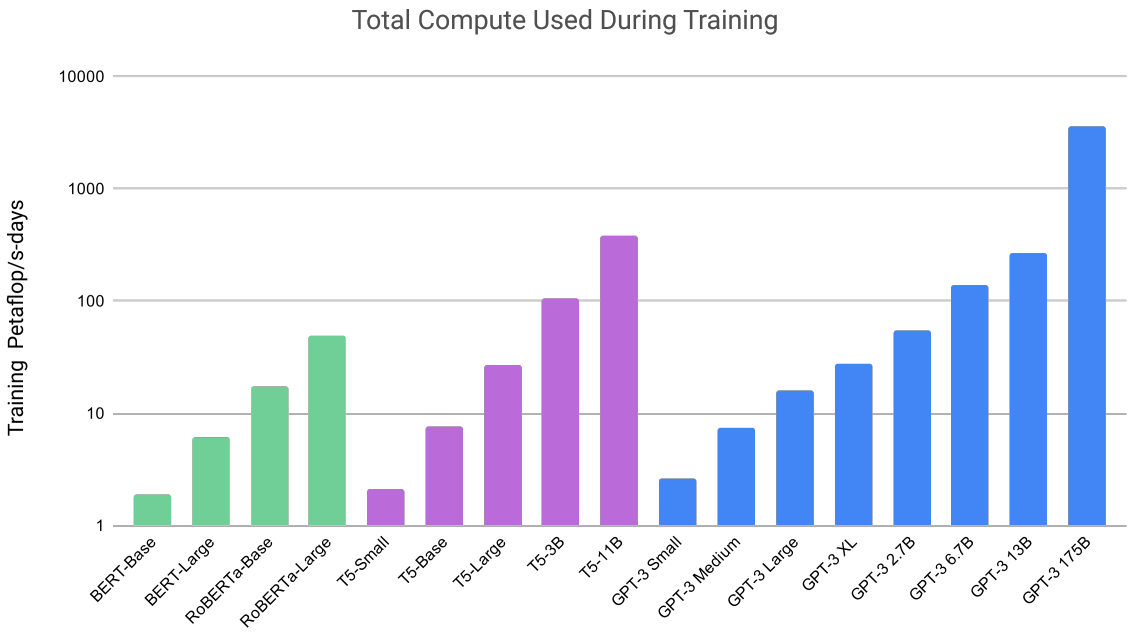


Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH+20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

A major methodological concern with language models pretrained on a broad swath of internet data, particularly large models with the capacity to memorize vast amounts of content, is potential contamination of downstream tasks by having their test or development sets inadvertently seen during pre-training. To reduce such contamination, we searched for and attempted to remove any overlaps with the development and test sets of all benchmarks studied in this paper. Unfortunately, a bug in the filtering caused us to ignore some overlaps, and due to the cost of training it was not feasible to retrain the model. In Section 4 we characterize the impact of the remaining overlaps, and in future work we will more aggressively remove data contamination.

2.3 Training Process

As found in [KMH+20, MKAT18], larger models can typically use a larger batch size, but require a smaller learning rate. We measure the gradient noise scale during training and use it to guide our choice of batch size [MKAT18]. Table 2.1 shows the parameter settings we used. To train the larger models without running out of memory, we use a mixture of model parallelism within each matrix multiply and model parallelism across the layers of the network. All models were trained on V100 GPU's on part of a high-bandwidth cluster provided by Microsoft. Details of the training process and hyperparameter settings are described in Appendix B.

2.4 Evaluation

For few-shot learning, we evaluate each example in the evaluation set by randomly drawing K examples from that task's training set as conditioning, delimited by 1 or 2 newlines depending on the task. For LAMBADA and Storycloze there is no supervised training set available so we draw conditioning examples from the development set and evaluate on the test set. For Winograd (the original, not SuperGLUE version) there is only one dataset, so we draw conditioning examples directly from it.

K can be any value from 0 to the maximum amount allowed by the model's context window, which is $n_{\text{ctx}} = 2048$ for all models and typically fits 10 to 100 examples. Larger values of K are usually but not always better, so when a separate development and test set are available, we experiment with a few values of K on the development set and then run the best value on the test set. For some tasks (see Appendix G) we also use a natural language prompt in addition to (or for $K = 0$, instead of) demonstrations.

On tasks that involve choosing one correct completion from several options (multiple choice), we provide K examples of context plus correct completion, followed by one example of context only, and compare the LM likelihood of each completion. For most tasks we compare the per-token likelihood (to normalize for length), however on a small number of datasets (ARC, OpenBookQA, and RACE) we gain additional benefit as measured on the development set by normalizing by the unconditional probability of each completion, by computing $\frac{P(\text{completion}|\text{context})}{P(\text{completion}|\text{answer\_context})}$, where answer_context is the string `"Answer: "` or `"A: "` and is used to prompt that the completion should be an answer but is otherwise generic.

On tasks that involve binary classification, we give the options more semantically meaningful names (e.g. "True" or "False" rather than 0 or 1) and then treat the task like multiple choice; we also sometimes frame the task similar to what is done by [RSR+19] (see Appendix G) for details.

On tasks with free-form completion, we use beam search with the same parameters as [RSR+19]: a beam width of 4 and a length penalty of $\alpha = 0.6$. We score the model using F1 similarity score, BLEU, or exact match, depending on what is standard for the dataset at hand.

Final results are reported on the test set when publicly available, for each model size and learning setting (zero-, one-, and few-shot). When the test set is private, our model is often too large to fit on the test server, so we report results on the development set. We do submit to the test server on a small number of datasets (SuperGLUE, TriviaQA, PiQa) where we were able to make submission work, and we submit only the 200B few-shot results, and report development set results for everything else.

3 Results

In Figure 3.1 we display training curves for the 8 models described in Section 2. For this graph we also include 6 additional extra-small models with as few as 100,000 parameters. As observed in [KMH+20], language modeling performance follows a power-law when making efficient use of training compute. After extending this trend by two more orders of magnitude, we observe only a slight (if any) departure from the power-law. One might worry that these improvements in cross-entropy loss come only from modeling spurious details of our training corpus. However, we will see in the following sections that improvements in cross-entropy loss lead to consistent performance gains across a broad spectrum of natural language tasks.

Below, we evaluate the 8 models described in Section 2 (the 175 billion parameter parameter GPT-3 and 7 smaller models) on a wide range of datasets. We group the datasets into 9 categories representing roughly similar tasks.

In Section 3.1 we evaluate on traditional language modeling tasks and tasks that are similar to language modeling, such as Cloze tasks and sentence/paragraph completion tasks. In Section 3.2 we evaluate on "closed book" question answering tasks: tasks which require using the information stored in the model's parameters to answer general knowledge questions. In Section 3.3 we evaluate the model's ability to translate between languages (especially one-shot and few-shot). In Section 3.4 we evaluate the model's performance on Winograd Schema-like tasks. In Section 3.5 we evaluate on datasets that involve commonsense reasoning or question answering. In Section 3.6 we evaluate on reading comprehension tasks, in Section 3.7 we evaluate on the SuperGLUE benchmark suite, and in 3.8 we briefly explore NLI. Finally, in Section 3.9, we invent some additional tasks designed especially to probe in-context learning abilities – these tasks focus on on-the-fly reasoning, adaptation skills, or open-ended text synthesis. We evaluate all tasks in the few-shot, one-shot, and zero-shot settings.

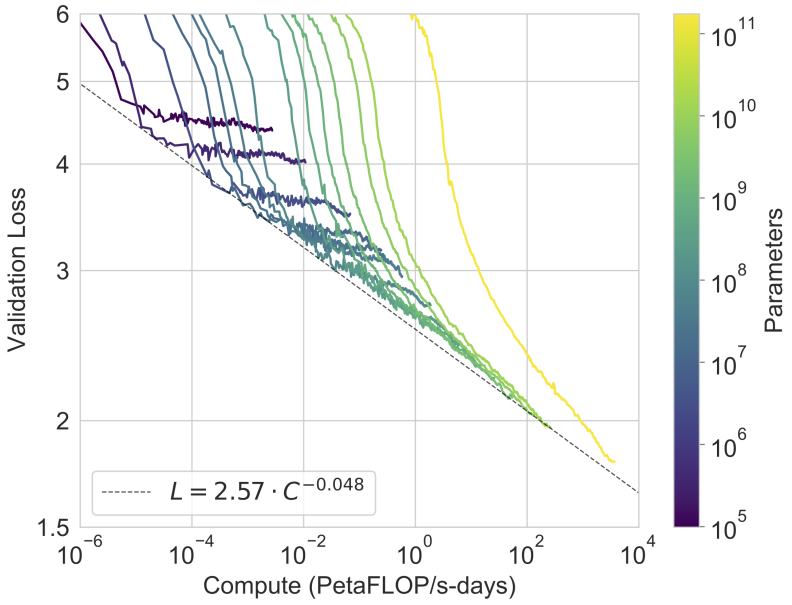


Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH+20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

Table 3.1: Zero-shot results on PTB language modeling dataset. Many other common language modeling datasets are omitted because they are derived from Wikipedia or other sources which are included in GPT-3's training data. [a][RWC+19]

3.1 Language Modeling, Cloze, and Completion Tasks

In this section we test GPT-3's performance on the traditional task of language modeling, as well as related tasks that involve predicting a single word of interest, completing a sentence or paragraph, or choosing between possible completions of a piece of text.

3.1.1 Language Modeling

We calculate zero-shot perplexity on the Penn Tree Bank (PTB) [MKM+94] dataset measured in [RWC+19]. We omit the 4 Wikipedia-related tasks in that work because they are entirely contained in our training data, and we also omit the one-billion word benchmark due to a high fraction of the dataset being contained in our training set. PTB escapes these issues due to predating the modern internet. Our largest model sets a new SOTA on PTB by a substantial margin of 15 points, achieving a perplexity of 20.50. Note that since PTB is a traditional language modeling dataset it does not have a clear separation of examples to define one-shot or few-shot evaluation around, so we measure only zero-shot.

3.1.2 LAMBADA

The LAMBADA dataset [PKL+16] tests the modeling of long-range dependencies in text – the model is asked to predict the last word of sentences which require reading a paragraph of context. It has recently been suggested that the continued scaling of language models is yielding diminishing returns on this difficult benchmark. [BHT+20] reflect on the small 1.5% improvement achieved by a doubling of model size between two recent state of the art results ([SPP+19]

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Table 3.2: Performance on cloze and completion tasks. GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets. [a][Tur20] [b][RWC+19] [c][LDL19] [d][LCH+20]

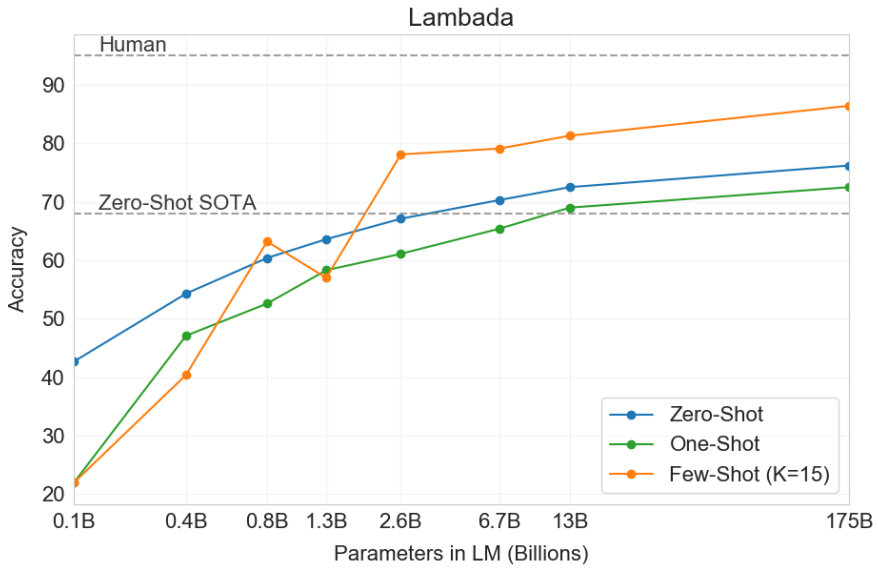


Figure 3.2: On LAMBADA, the few-shot capability of language models results in a strong boost to accuracy. GPT-3 2.7B outperforms the SOTA 17B parameter Turing-NLG [Tur20] in this setting, and GPT-3 175B advances the state of the art by 18%. Note zero-shot uses a different format from one-shot and few-shot as described in the text.

and [Tur20]) and argue that "continuing to expand hardware and data sizes by orders of magnitude is not the path forward". We find that path is still promising and in a zero-shot setting GPT-3 achieves 76% on LAMBADA, a gain of 8% over the previous state of the art.

LAMBADA is also a demonstration of the flexibility of few-shot learning as it provides a way to address a problem that classically occurs with this dataset. Although the completion in LAMBADA is always the last word in a sentence, a standard language model has no way of knowing this detail. It thus assigns probability not only to the correct ending but also to other valid continuations of the paragraph. This problem has been partially addressed in the past with stop-word filters [RWC+19] (which ban "continuation" words). The few-shot setting instead allows us to "frame" the task as a cloze-test and allows the language model to infer from examples that a completion of exactly one word is desired. We use the following fill-in-the-blank format:

Alice was friends with Bob. Alice went to visit her friend _____. → Bob

George bought some baseball equipment, a ball, a glove, and a _____. →

When presented with examples formatted this way, GPT-3 achieves 86.4% accuracy in the few-shot setting, an increase of over 18% from the previous state-of-the-art. We observe that few-shot performance improves strongly with model size. While this setting decreases the performance of the smallest model by almost 20%, for GPT-3 it improves accuracy by 10%. Finally, the fill-in-blank method is not effective one-shot, where it always performs worse than the zero-shot setting. Perhaps this is because all models still require several examples to recognize the pattern.

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Table 3.3: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

One note of caution is that an analysis of test set contamination identified that a significant minority of the LAMBADA dataset appears to be present in our training data – however analysis performed in Section 4 suggests negligible impact on performance.

3.1.3 HellaSwag

The HellaSwag dataset [ZHB+19] involves picking the best ending to a story or set of instructions. The examples were adversarially mined to be difficult for language models while remaining easy for humans (who achieve 95.6% accuracy). GPT-3 achieves 78.1% accuracy in the one-shot setting and 79.3% accuracy in the few-shot setting, outperforming the 75.4% accuracy of a fine-tuned 1.5B parameter language model [ZHR+19] but still a fair amount lower than the overall SOTA of 85.6% achieved by the fine-tuned multi-task model ALUM.

3.1.4 StoryCloze

We next evaluate GPT-3 on the StoryCloze 2016 dataset [MCH+16], which involves selecting the correct ending sentence for five-sentence long stories. Here GPT-3 achieves 83.2% in the zero-shot setting and 87.7% in the few-shot setting (with $K = 70$). This is still 4.1% lower than the fine-tuned SOTA using a BERT based model [LDL19] but improves over previous zero-shot results by roughly 10%.

3.2 Closed Book Question Answering

In this section we measure GPT-3's ability to answer questions about broad factual knowledge. Due to the immense amount of possible queries, this task has normally been approached by using an information retrieval system to find relevant text in combination with a model which learns to generate an answer given the question and the retrieved text. Since this setting allows a system to search for and condition on text which potentially contains the answer it is denoted "open-book". [RRS20] recently demonstrated that a large language model can perform surprisingly well directly answering the questions without conditioning on auxilliary information. They denote this more restrictive evaluation setting as "closed-book". Their work suggests that even higher-capacity models could perform even better and we test this hypothesis with GPT-3. We evaluate GPT-3 on the 3 datasets in [RRS20]: Natural Questions [KPR+19], WebQuestions [BCFL13], and TriviaQA [JCWZ17], using the same splits. Note that in addition to all results being in the closed-book setting, our use of few-shot, one-shot, and zero-shot evaluations represent an even stricter setting than previous closed-book QA work: in addition to external content not being allowed, fine-tuning on the Q&A dataset itself is also not permitted.

The results for GPT-3 are shown in Table 3.3. On TriviaQA, we achieve 64.3% in the zero-shot setting, 68.0% in the one-shot setting, and 71.2% in the few-shot setting. The zero-shot result already outperforms the fine-tuned T5-11B by 14.2%, and also outperforms a version with Q&A tailored span prediction during pre-training by 3.8%. The one-shot result improves by 3.7% and matches the SOTA for an open-domain QA system which not only fine-tunes but also makes use of a learned retrieval mechanism over a 15.3B parameter dense vector index of 21M documents [LPP+20]. GPT-3's few-shot result further improves performance another 3.2% beyond this.

On WebQuestions (WebQs), GPT-3 achieves 14.4% in the zero-shot setting, 25.3% in the one-shot setting, and 41.5% in the few-shot setting. This compares to 37.4% for fine-tuned T5-11B, and 44.7% for fine-tuned T5-11B+SSM, which uses a Q&A-specific pre-training procedure. GPT-3 in the few-shot setting approaches the performance of state-of-the-art fine-tuned models. Notably, compared to TriviaQA, WebQS shows a much larger gain from zero-shot to few-shot (and indeed its zero-shot and one-shot performance are poor), perhaps suggesting that the WebQs questions

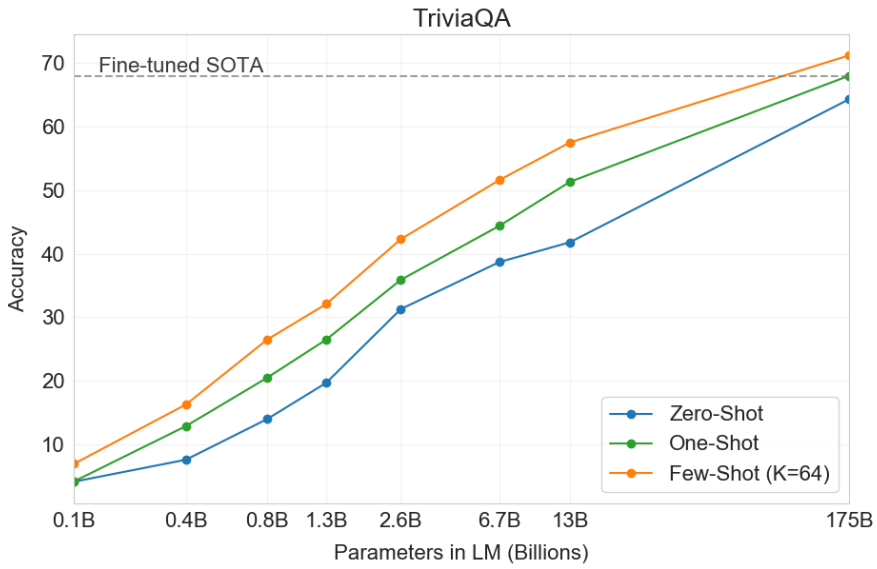


Figure 3.3: On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

and/or the style of their answers are out-of-distribution for GPT-3. Nevertheless, GPT-3 appears able to adapt to this distribution, recovering strong performance in the few-shot setting.

On Natural Questions (NQs) GPT-3 achieves 14.6% in the zero-shot setting, 23.0% in the one-shot setting, and 29.9% in the few-shot setting, compared to 36.6% for fine-tuned T5 11B+SSM. Similar to WebQS, the large gain from zero-shot to few-shot may suggest a distribution shift, and may also explain the less competitive performance compared to TriviaQA and WebQS. In particular, the questions in NQs tend towards very fine-grained knowledge on Wikipedia specifically which could be testing the limits of GPT-3's capacity and broad pretraining distribution.

Overall, on one of the three datasets GPT-3's one-shot matches the open-domain fine-tuning SOTA. On the other two datasets it approaches the performance of the closed-book SOTA despite not using fine-tuning. On all 3 datasets, we find that performance scales very smoothly with model size (Figure 3.3 and Appendix H Figure H.7), possibly reflecting the idea that model capacity translates directly to more 'knowledge' absorbed in the parameters of the model.

3.3 Translation

For GPT-2 a filter was used on a multilingual collection of documents to produce an English only dataset due to capacity concerns. Even with this filtering GPT-2 showed some evidence of multilingual capability and performed non-trivially when translating between French and English despite only training on 10 megabytes of remaining French text. Since we increase the capacity by over two orders of magnitude from GPT-2 to GPT-3, we also expand the scope of the training dataset to include more representation of other languages, though this remains an area for further improvement. As discussed in 2.2 the majority of our data is derived from raw Common Crawl with only quality-based filtering. Although GPT-3's training data is still primarily English (93% by word count), it also includes 7% of text in other languages. These languages are documented in the supplemental material. In order to better understand translation capability, we also expand our analysis to include two additional commonly studied languages, German and Romanian.

Existing unsupervised machine translation approaches often combine pretraining on a pair of monolingual datasets with back-translation [SHB15] to bridge the two languages in a controlled way. By contrast, GPT-3 learns from a blend of training data that mixes many languages together in a natural way, combining them on a word, sentence, and document level. GPT-3 also uses a single training objective which is not customized or designed for any task in particular. However, our one / few-shot settings aren't strictly comparable to prior unsupervised work since they make use of a small amount of paired examples (1 or 64). This corresponds to up to a page or two of in-context training data.

Results are shown in Table 3.4. Zero-shot GPT-3, which only receives on a natural language description of the task, still underperforms recent unsupervised NMT results. However, providing only a single example demonstration for

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

Table 3.4: Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English reflecting its strength as an English LM. We report BLEU scores on the WMT'14 Fr↔En, WMT'16 De↔En, and WMT'16 Ro↔En datasets as measured by `multi-bleu.perl` with XLM's tokenization in order to compare most closely with prior unsupervised NMT work. SacreBLEU[f] [Pos18] results reported in Appendix H. Underline indicates an unsupervised or few-shot SOTA, bold indicates supervised SOTA with relative confidence. [a][EOAG18] [b][DHKH14] [c][WXH+18] [d][oR16] [e][LGG+20] [f][SacreBLEU signature: BLEU+case.mixed+numrefs.1+smooth.exp+tok.intl+version.1.2.20]

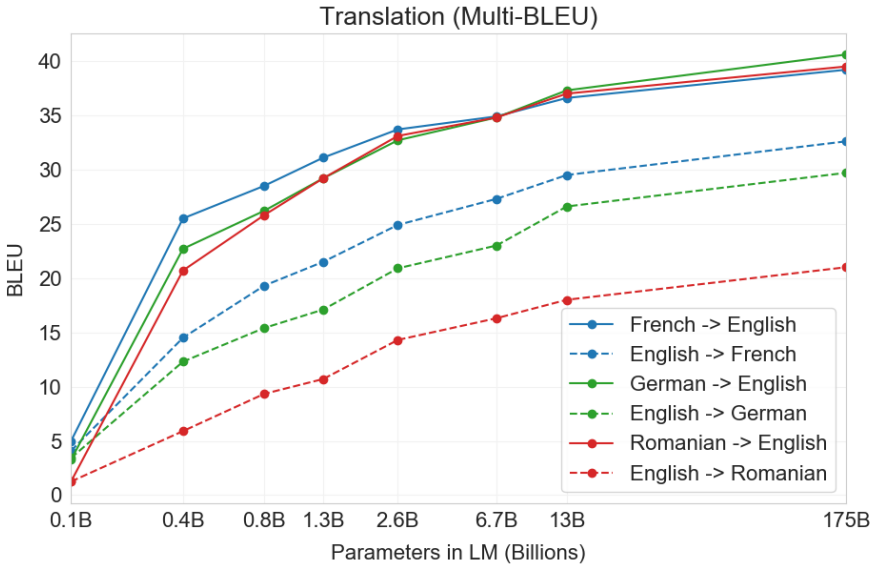


Figure 3.4: Few-shot translation performance on 6 language pairs as model capacity increases. There is a consistent trend of improvement across all datasets as the model scales, and as well as tendency for translation into English to be stronger than translation from English.

Setting	Winograd	Winogrande (XL)
Fine-tuned SOTA	**90.1**[a]	**84.6**[b]
GPT-3 Zero-Shot	88.3*	70.2
GPT-3 One-Shot	89.7*	73.2
GPT-3 Few-Shot	88.6*	77.7

Table 3.5: Results on the WSC273 version of Winograd schemas and the adversarial Winogrande dataset. See Section 4 for details on potential contamination of the Winograd test set. [a][SBBC19] [b][LYN+20]

Figure 3.5: Zero-, one-, and few-shot performance on the adversarial Winogrande dataset as model capacity scales. Scaling is relatively smooth with the gains to few-shot learning increasing with model size, and few-shot GPT-3 175B is competitive with a fine-tuned RoBERTA-large.

each translation task improves performance by over 7 BLEU and nears competitive performance with prior work. GPT-3 in the full few-shot setting further improves another 4 BLEU resulting in similar average performance to prior unsupervised NMT work. GPT-3 has a noticeable skew in its performance depending on language direction. For the three input languages studied, GPT-3 significantly outperforms prior unsupervised NMT work when translating into English but underperforms when translating in the other direction. Performance on En-Ro is a noticeable outlier at over 10 BLEU worse than prior unsupervised NMT work. This could be a weakness due to reusing the byte-level BPE tokenizer of GPT-2 which was developed for an almost entirely English training dataset. For both Fr-En and De-En, few shot GPT-3 outperforms the best supervised result we could find but due to our unfamiliarity with the literature and the appearance that these are un-competitive benchmarks we do not suspect those results represent true state of the art. For Ro-En, few shot GPT-3 performs within 0.5 BLEU of the overall SOTA which is achieved by a combination of unsupervised pretraining, supervised finetuning on 608K labeled examples, and backtranslation [LHCG19b].

Finally, across all language pairs and across all three settings (zero-, one-, and few-shot), there is a smooth trend of improvement with model capacity. This is shown in Figure 3.4 in the case of few-shot results, and scaling for all three settings is shown in Appendix H.

3.4 Winograd-Style Tasks

The Winograd Schemas Challenge [LDM12] is a classical task in NLP that involves determining which word a pronoun refers to, when the pronoun is grammatically ambiguous but semantically unambiguous to a human. Recently fine-tuned language models have achieved near-human performance on the original Winograd dataset, but more difficult versions

Setting	PIQA	ARC (Easy)	ARC (Challenge)	OpenBookQA
Fine-tuned SOTA	79.4	**92.0**[KKS+20]	**78.5**[KKS+20]	**87.2**[KKS+20]
GPT-3 Zero-Shot	**80.5***	68.8	51.4	57.6
GPT-3 One-Shot	**80.5***	71.2	53.2	58.8
GPT-3 Few-Shot	**82.8***	70.1	51.5	65.4

Table 3.6: GPT-3 results on three commonsense reasoning tasks, PIQA, ARC, and OpenBookQA. GPT-3 Few-Shot PIQA result is evaluated on the test server. See Section 4 for details on potential contamination issues on the PIQA test set.

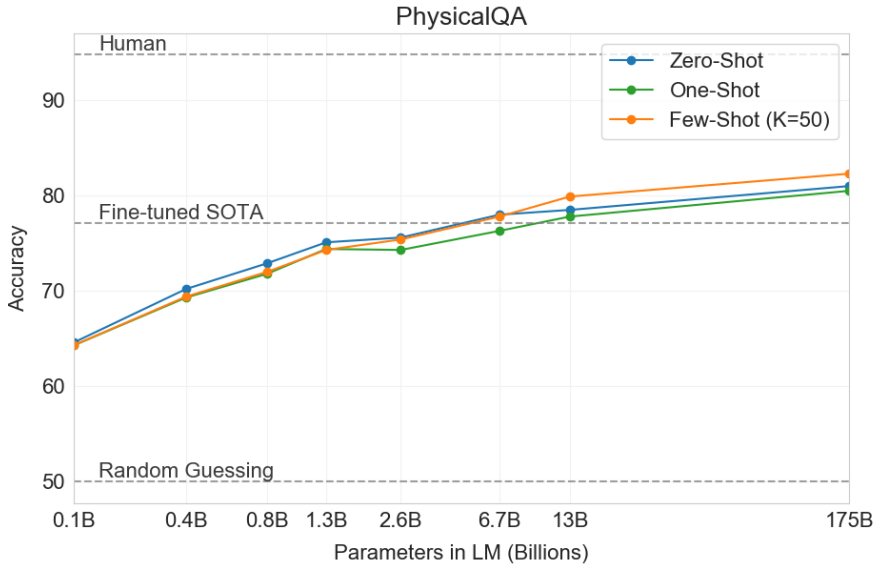


Figure 3.6: GPT-3 results on PIQA in the zero-shot, one-shot, and few-shot settings. The largest model achieves a score on the development set in all three conditions that exceeds the best recorded score on the task.

such as the adversarially-mined Winogrande dataset [SBBC19] still significantly lag human performance. We test GPT-3's performance on both Winograd and Winogrande, as usual in the zero-, one-, and few-shot setting.

On Winograd we test GPT-3 on the original set of 273 Winograd schemas, using the same "partial evaluation" method described in [RWC+19]. Note that this setting differs slightly from the WSC task in the SuperGLUE benchmark, which is presented as binary classification and requires entity extraction to convert to the form described in this section. On Winograd GPT-3 achieves 88.3%, 89.7%, and 88.6% in the zero-shot, one-shot, and few-shot settings, showing no clear in-context learning but in all cases achieving strong results just a few points below state-of-the-art and estimated human performance. We note that contamination analysis found some Winograd schemas in the training data but this appears to have only a small effect on results (see Section 4).

On the more difficult Winogrande dataset, we do find gains to in-context learning: GPT-3 achieves 70.2% in the zero-shot setting, 73.2% in the one-shot setting, and 77.7% in the few-shot setting. For comparison a fine-tuned RoBERTA model achieves 79%, state-of-the-art is 84.6% achieved with a fine-tuned high capacity model (T5), and human performance on the task as reported by [SBBC19] is 94.0%.

3.5 Common Sense Reasoning

Next we consider three datasets which attempt to capture physical or scientific reasoning, as distinct from sentence completion, reading comprehension, or broad knowledge question answering. The first, PhysicalQA (PIQA) [BZB+19], asks common sense questions about how the physical world works and is intended as a probe of grounded understanding of the world. GPT-3 achieves 81.0% accuracy zero-shot, 80.5% accuracy one-shot, and 82.8% accuracy few-shot (the last measured on PIQA's test server). This compares favorably to the 79.4% accuracy prior state-of-the-art of a

Setting	CoQA	DROP	QuAC	SQuADv2	RACE-h	RACE-m
Fine-tuned SOTA	**90.7**[a]	**89.1**[b]	**74.4**[c]	**93.0**[d]	**90.0**[e]	**93.1**[e]
GPT-3 Zero-Shot	81.5	23.6	41.5	59.5	45.5	58.4
GPT-3 One-Shot	84.0	34.3	43.3	65.4	45.9	57.4
GPT-3 Few-Shot	85.0	36.5	44.3	69.8	46.8	58.1

Table 3.7: Results on reading comprehension tasks. All scores are F1 except results for RACE which report accuracy. [a][JZC+19] [b][JN20] [c][AI19] [d][QIA20] [e][SPP+19]

fine-tuned RoBERTa. PIQA shows relatively shallow scaling with model size and is still over 10% worse than human performance, but GPT-3's few-shot and even zero-shot result outperform the current state-of-the-art. Our analysis flagged PIQA for a potential data contamination issue (despite hidden test labels), and we therefore conservatively mark the result with an asterisk. See Section 4 for details.

ARC [CCE+18] is a dataset of multiple-choice questions collected from 3rd to 9th grade science exams. On the "Challenge" version of the dataset which has been filtered to questions which simple statistical or information retrieval methods are unable to correctly answer, GPT-3 achieves 51.4% accuracy in the zero-shot setting, 53.2% in the one-shot setting, and 51.5% in the few-shot setting. This is approaching the performance of a fine-tuned RoBERTa baseline (55.9%) from UnifiedQA [KKS+20]. On the "Easy" version of the dataset (questions which either of the mentioned baseline approaches answered correctly), GPT-3 achieves 68.8%, 71.2%, and 70.1% which slightly exceeds a fine-tuned RoBERTa baseline from [KKS+20]. However, both of these results are still much worse than the overall SOTAs achieved by the UnifiedQA which exceeds GPT-3's few-shot results by 27% on the challenge set and 22% on the easy set.

On OpenBookQA [MCKS18], GPT-3 improves significantly from zero to few shot settings but is still over 20 points short of the overall SOTA. GPT-3's few-shot performance is similar to a fine-tuned BERT Large baseline on the leaderboard.

Overall, in-context learning with GPT-3 shows mixed results on commonsense reasoning tasks, with only small and inconsistent gains observed in the one and few-shot learning settings for both PIQA and ARC, but a significant improvement is observed on OpenBookQA. GPT-3 sets SOTA on the new PIQA dataset in all evaluation settings.

3.6 Reading Comprehension

Next we evaluate GPT-3 on the task of reading comprehension. We use a suite of 5 datasets including abstractive, multiple choice, and span based answer formats in both dialog and single question settings. We observe a wide spread in GPT-3's performance across these datasets suggestive of varying capability with different answer formats. In general we observe GPT-3 is on par with initial baselines and early results trained using contextual representations on each respective dataset.

GPT-3 performs best (within 3 points of the human baseline) on CoQA [RCM19] a free-form conversational dataset and performs worst (13 F1 below an ELMo baseline) on QuAC [CHI+18] a dataset which requires modeling structured dialog acts and answer span selections of teacher-student interactions. On DROP [DWD+19], a dataset testing discrete reasoning and numeracy in the context of reading comprehension, GPT-3 in a few-shot setting outperforms the fine-tuned BERT baseline from the original paper but is still well below both human performance and state-of-the-art approaches which augment neural networks with symbolic systems [RLL+19]. On SQuAD 2.0 [RJL18], GPT-3 demonstrates its few-shot learning capabilities, improving by almost 10 F1 (to 69.8) compared to a zero-shot setting. This allows it to slightly outperform the best fine-tuned result in the original paper. On RACE [LXL+17], a multiple choice dataset of middle school and high school english examinations, GPT-3 performs relatively weakly and is only competitive with the earliest work utilizing contextual representations and is still 45% behind SOTA.

3.7 SuperGLUE

In order to better aggregate results on NLP tasks and compare to popular models such as BERT and RoBERTa in a more systematic way, we also evaluate GPT-3 on a standardized collection of datasets, the SuperGLUE benchmark [WPN+19] [WPN+19] [CLC+19] [DMST19] [RBG11] [KCR+18] [ZLL+18] [DGM06] [BHDD+06] [GMDD07] [BDD+09] [PCC18] [PHR+18]. GPT-3's test-set performance on the SuperGLUE dataset is shown in Table 3.8. In the few-shot setting, we used 32 examples for all tasks, sampled randomly from the training set. For all tasks except WSC

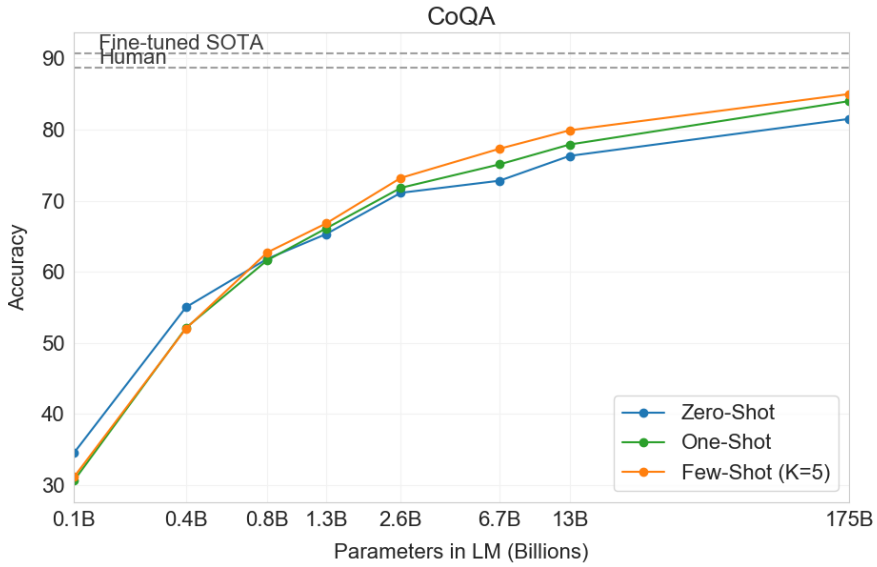


Figure 3.7: GPT-3 results on CoQA reading comprehension task. GPT-3 175B achieves 85 F1 in the few-shot setting, only a few points behind measured human performance and state-of-the-art fine-tuned models. Zero-shot and one-shot performance is a few points behind, with the gains to few-shot being largest for bigger models.

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

Table 3.8: Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

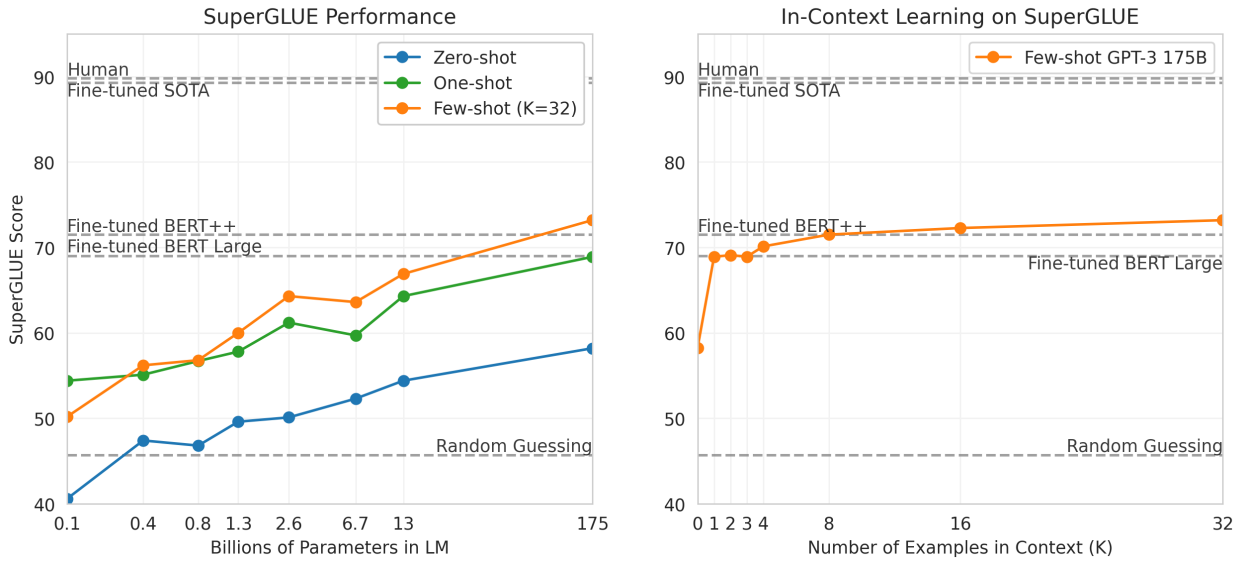


Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context. A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not directly comparable to the dotted reference lines (our test set results are in Table 3.8). The BERT-Large reference model was fine-tuned on the SuperGLUE training set (125K examples), whereas BERT++ was first fine-tuned on MultiNLI (392K examples) and SWAG (113K examples) before further fine-tuning on the SuperGLUE training set (for a total of 630K fine-tuning examples). We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

and MultiRC, we sampled a new set of examples to use in the context for each problem. For WSC and MultiRC, we used the same set of randomly drawn examples from the training set as context for all of the problems we evaluated.

We observe a wide range in GPT-3's performance across tasks. On COPA and ReCoRD GPT-3 achieves near-SOTA performance in the one-shot and few-shot settings, with COPA falling only a couple points short and achieving second place on the leaderboard, where first place is held by a fine-tuned 11 billion parameter model (T5). On WSC, performance is still relatively strong, achieving 80.1% in the few-shot setting (note that GPT-3 achieves 88.6% on the original Winograd dataset as described in Section 3.4). On BoolQ, MultiRC, and RTE, performance is reasonable, roughly matching that of a fine-tuned BERT-Large. On CB, we see signs of life at 75.6% in the few-shot setting.

WiC is a notable weak spot with few-shot performance at 49.4% (at random chance). We tried a number of different phrasings and formulations for WiC (which involves determining if a word is being used with the same meaning in two sentences), none of which was able to achieve strong performance. This hints at a phenomenon that will become clearer in the next section (which discusses the ANLI benchmark) – GPT-3 appears to be weak in the few-shot or one-shot setting at some tasks that involve comparing two sentences or snippets, for example whether a word is used the same way in two sentences (WiC), whether one sentence is a paraphrase of another, or whether one sentence implies another. This could also explain the comparatively low scores for RTE and CB, which also follow this format. Despite these weaknesses, GPT-3 still outperforms a fine-tuned BERT-large on four of eight tasks and on two tasks GPT-3 is close to the state-of-the-art held by a fine-tuned 11 billion parameter model.

Finally, we note that the few-shot SuperGLUE score steadily improves with both model size and with number of examples in the context showing increasing benefits from in-context learning (Figure 3.8). We scale K up to 32 examples per task, after which point additional examples will not reliably fit into our context. When sweeping over values of K, we find that GPT-3 requires less than eight total examples per task to outperform a fine-tuned BERT-Large on overall SuperGLUE score.

3.8 NLI

Natural Language Inference (NLI) [Fyo00] concerns the ability to understand the relationship between two sentences. In practice, this task is usually structured as a two or three class classification problem where the model classifies

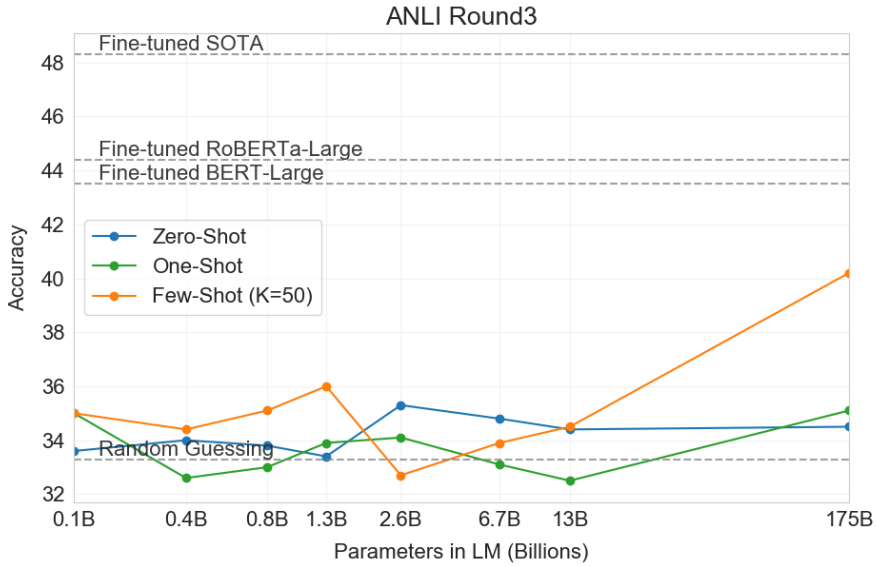


Figure 3.9: Performance of GPT-3 on ANLI Round 3. Results are on the dev-set, which has only 1500 examples and therefore has high variance (we estimate a standard deviation of 1.2%). We find that smaller models hover around random chance, while few-shot GPT-3 175B closes almost half the gap from random chance to SOTA. Results for ANLI rounds 1 and 2 are shown in the appendix.

whether the second sentence logically follows from the first, contradicts the first sentence, or is possibly true (neutral). SuperGLUE includes an NLI dataset, RTE, which evaluates the binary version of the task. On RTE, only the largest version of GPT-3 performs convincingly better than random (56%) in any evaluation setting, but in a few-shot setting GPT-3 performs similarly to a single-task fine-tuned BERT Large. We also evaluate on the recently introduced Adversarial Natural Language Inference (ANLI) dataset [NWD+19]. ANLI is a difficult dataset employing a series of adversarially mined natural language inference questions in three rounds (R1, R2, and R3). Similar to RTE, all of our models smaller than GPT-3 perform at almost exactly random chance on ANLI, even in the few-shot setting ($\sim 33\%$), whereas GPT-3 itself shows signs of life on Round 3. Results for ANLI R3 are highlighted in Figure 3.9 and full results for all rounds can be found in Appendix H. These results on both RTE and ANLI suggest that NLI is still a very difficult task for language models and they are only just beginning to show signs of progress.

3.9 Synthetic and Qualitative Tasks

One way to probe GPT-3's range of abilities in the few-shot (or zero- and one-shot) setting is to give it tasks which require it to perform simple on-the-fly computational reasoning, recognize a novel pattern that is unlikely to have occurred in training, or adapt quickly to an unusual task. We devise several tasks to test this class of abilities. First, we test GPT-3's ability to perform arithmetic. Second, we create several tasks that involve rearranging or unscrambling the letters in a word, tasks which are unlikely to have been exactly seen during training. Third, we test GPT-3's ability to solve SAT-style analogy problems few-shot. Finally, we test GPT-3 on several qualitative tasks, including using new words in a sentence, correcting English grammar, and news article generation. We will release the synthetic datasets with the hope of stimulating further study of test-time behavior of language models.

3.9.1 Arithmetic

To test GPT-3's ability to perform simple arithmetic operations without task-specific training, we developed a small battery of 10 tests that involve asking GPT-3 a simple arithmetic problem in natural language:

- **2 digit addition (2D+)** – The model is asked to add two integers sampled uniformly from $[0, 100)$, phrased in the form of a question, e.g. "Q: What is 48 plus 76? A: 124."
- **2 digit subtraction (2D-)** – The model is asked to subtract two integers sampled uniformly from $[0, 100)$; the answer may be negative. Example: "Q: What is 34 minus 53? A: -19".
- **3 digit addition (3D+)** – Same as 2 digit addition, except numbers are uniformly sampled from $[0, 1000)$.

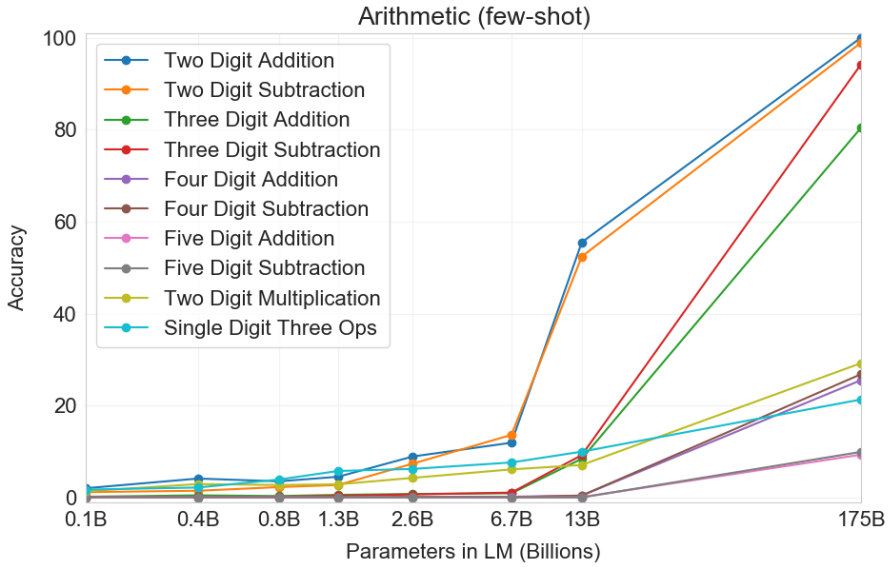


Figure 3.10: Results on all 10 arithmetic tasks in the few-shot settings for models of different sizes. There is a significant jump from the second largest model (GPT-3 13B) to the largest model (GPT-3 175), with the latter being able to reliably accurate 2 digit arithmetic, usually accurate 3 digit arithmetic, and correct answers a significant fraction of the time on 4-5 digit arithmetic, 2 digit multiplication, and compound operations. Results for one-shot and zero-shot are shown in the appendix.

- **3 digit subtraction (3D-)** – Same as 2 digit subtraction, except numbers are uniformly sampled from $[0, 1000)$.
- **4 digit addition (4D+)** – Same as 3 digit addition, except uniformly sampled from $[0, 10000)$.
- **4 digit subtraction (4D-)** – Same as 3 digit subtraction, except uniformly sampled from $[0, 10000)$.
- **5 digit addition (5D+)** – Same as 3 digit addition, except uniformly sampled from $[0, 100000)$.
- **5 digit subtraction (5D-)** – Same as 3 digit subtraction, except uniformly sampled from $[0, 100000)$.
- **2 digit multiplication (2Dx)** – The model is asked to multiply two integers sampled uniformly from $[0, 100)$, e.g. "Q: What is 24 times 42? A: 1008".
- **One-digit composite (1DC)** – The model is asked to perform a composite operation on three 1 digit numbers, with parentheses around the last two. For example, "Q: What is 6+(4*8)? A: 38". The three 1 digit numbers are selected uniformly on $[0, 10)$ and the operations are selected uniformly from {+,-,*}.

In all 10 tasks the model must generate the correct answer exactly. For each task we generate a dataset of 2,000 random instances of the task and evaluate all models on those instances.

First we evaluate GPT-3 in the few-shot setting, for which results are shown in Figure 3.10. On addition and subtraction, GPT-3 displays strong proficiency when the number of digits is small, achieving 100% accuracy on 2 digit addition, 98.9% at 2 digit subtraction, 80.2% at 3 digit addition, and 94.2% at 3-digit subtraction. Performance decreases as the number of digits increases, but GPT-3 still achieves 25-26% accuracy on four digit operations and 9-10% accuracy on five digit operations, suggesting at least some capacity to generalize to larger numbers of digits. GPT-3 also achieves 29.2% accuracy at 2 digit multiplication, an especially computationally intensive operation. Finally, GPT-3 achieves 21.3% accuracy at single digit combined operations (for example, 9*(7+5)), suggesting that it has some robustness beyond just single operations.

As Figure 3.10 makes clear, small models do poorly on all of these tasks – even the 13 billion parameter model (the second largest after the 175 billion full GPT-3) can solve 2 digit addition and subtraction only half the time, and all other operations less than 10% of the time.

One-shot and zero-shot performance are somewhat degraded relative to few-shot performance, suggesting that adaptation to the task (or at the very least recognition of the task) is important to performing these computations correctly. Nevertheless, one-shot performance is still quite strong, and even zero-shot performance of the full GPT-3 significantly

Setting	2D+	2D-	3D+	3D-	4D+	4D-	5D+	5D-	2Dx	1DC
GPT-3 Zero-shot	76.9	58.0	34.2	48.3	4.0	7.5	0.7	0.8	19.8	9.8
GPT-3 One-shot	99.6	86.4	65.5	78.7	14.0	14.0	3.5	3.8	27.4	14.3
GPT-3 Few-shot	100.0	98.9	80.4	94.2	25.5	26.8	9.3	9.9	29.2	21.3

Table 3.9: Results on basic arithmetic tasks for GPT-3 175B. {2,3,4,5}D{+,-} is 2, 3, 4, and 5 digit addition or subtraction, 2Dx is 2 digit multiplication. 1DC is 1 digit composite operations. Results become progressively stronger moving from the zero-shot to one-shot to few-shot setting, but even the zero-shot shows significant arithmetic abilities.

Setting	CL	A1	A2	RI	RW
GPT-3 Zero-shot	3.66	2.28	8.91	8.26	0.09
GPT-3 One-shot	21.7	8.62	25.9	45.4	0.48
GPT-3 Few-shot	37.9	15.1	39.7	67.2	0.44

Table 3.10: GPT-3 175B performance on various word unscrambling and word manipulation tasks, in zero-, one-, and few-shot settings. CL is "cycle letters in word", A1 is anagrams of but the first and last letters, A2 is anagrams of all but the first and last two letters, RI is "Random insertion in word", RW is "reversed words".

outperforms few-shot learning for all smaller models. All three settings for the full GPT-3 are shown in Table 3.9, and model capacity scaling for all three settings is shown in Appendix H.

To spot-check whether the model is simply memorizing specific arithmetic problems, we took the 3-digit arithmetic problems in our test set and searched for them in our training data in both the forms `"<NUM1> + <NUM2> ="` and `"<NUM1> plus <NUM2>"`. Out of 2,000 addition problems we found only 17 matches (0.8%) and out of 2,000 subtraction problems we found only 2 matches (0.1%), suggesting that only a trivial fraction of the correct answers could have been memorized. In addition, inspection of incorrect answers reveals that the model often makes mistakes such as not carrying a "1", suggesting it is actually attempting to perform the relevant computation rather than memorizing a table.

Overall, GPT-3 displays reasonable proficiency at moderately complex arithmetic in few-shot, one-shot, and even zero-shot settings.

3.9.2 Word Scrambling and Manipulation Tasks

To test GPT-3's ability to learn novel symbolic manipulations from a few examples, we designed a small battery of 5 "character manipulation" tasks. Each task involves giving the model a word distorted by some combination of scrambling, addition, or deletion of characters, and asking it to recover the original word. The 5 tasks are:

- **Cycle letters in word (CL)** – The model is given a word with its letters cycled, then the "=" symbol, and is expected to generate the original word. For example, it might be given "lyinevitab" and should output "inevitably".
- **Anagrams of all but first and last characters (A1)** – The model is given a word where every letter except the first and last have been scrambled randomly, and must output the original word. Example: criroptuon = corruption.
- **Anagrams of all but first and last 2 characters (A2)** – The model is given a word where every letter except the first 2 and last 2 have been scrambled randomly, and must recover the original word. Example: opoepnnt $\rightarrow$ opponent.
- **Random insertion in word (RI)** – A random punctuation or space character is inserted between each letter of a word, and the model must output the original word. Example: s.u!c/c!e.s s i/o/n = succession.
- **Reversed words (RW)** – The model is given a word spelled backwards, and must output the original word. Example: stcejbo $\rightarrow$ objects.

For each task we generate 10,000 examples, which we chose to be the top 10,000 most frequent words as measured by [Nor09] of length more than 4 characters and less than 15 characters. The few-shot results are shown in Figure 3.11. Task performance tends to grow smoothly with model size, with the full GPT-3 model achieving 66.9% on removing

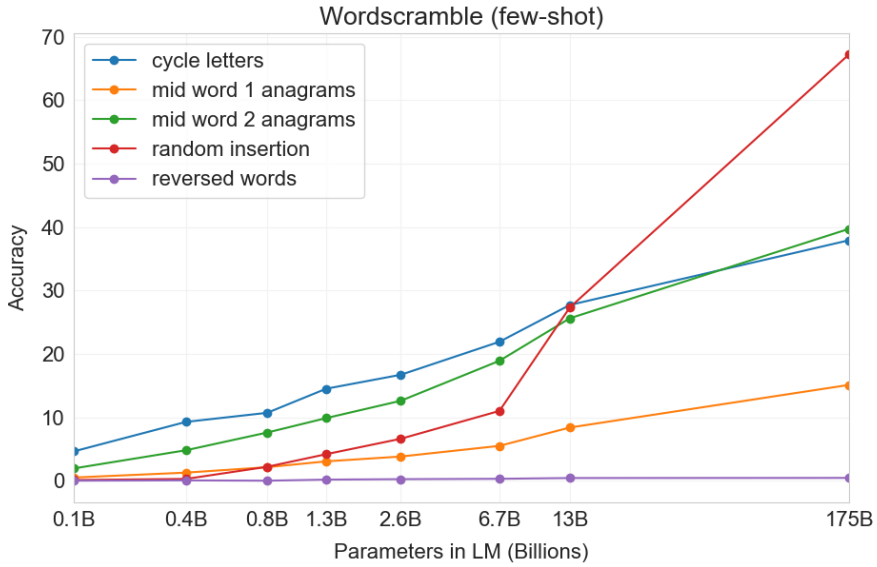


Figure 3.11: Few-shot performance on the five word scrambling tasks for different sizes of model. There is generally smooth improvement with model size although the random insertion task shows an upward slope of improvement with the 175B model solving the task the majority of the time. Scaling of one-shot and zero-shot performance is shown in the appendix. All tasks are done with $K = 100$.

random insertions, 38.6% on cycling letters, 40.2% on the easier anagram task, and 15.1% on the more difficult anagram task (where only the first and last letters are held fixed). None of the models can reverse the letters in a word.

In the one-shot setting, performance is significantly weaker (dropping by half or more), and in the zero-shot setting the model can rarely perform any of the tasks (Table 3.10). This suggests that the model really does appear to learn these tasks at test time, as the model cannot perform them zero-shot and their artificial nature makes them unlikely to appear in the pre-training data (although we cannot confirm this with certainty).

We can further quantify performance by plotting "in-context learning curves", which show task performance as a function of the number of in-context examples. We show in-context learning curves for the Symbol Insertion task in Figure 1.2. We can see that larger models are able to make increasingly effective use of in-context information, including both task examples and natural language task descriptions.

Finally, it is worth adding that solving these tasks requires character-level manipulations, whereas our BPE encoding operates on significant fractions of a word (on average ~ 0.7 words per token), so from the LM's perspective succeeding at these tasks involves not just manipulating BPE tokens but understanding and pulling apart their substructure. Also, CL, A1, and A2 are not bijective (that is, the unscrambled word is not a deterministic function of the scrambled word), requiring the model to perform some search to find the correct unscrambling. Thus, the skills involved appear to require non-trivial pattern-matching and computation.

3.9.3 SAT Analogies

To test GPT-3 on another task that is somewhat unusual relative to the typical distribution of text, we collected a set of 374 "SAT analogy" problems [TLBS03]. Analogies are a style of multiple choice question that constituted a section of the SAT college entrance exam before 2005. A typical example is "audacious is to boldness as (a) sanctimonious is to hypocrisy, (b) anonymous is to identity, (c) remorseful is to misdeed, (d) deleterious is to result, (e) impressionable is to temptation". The student is expected to choose which of the five word pairs has the same relationship as the original word pair; in this example the answer is "sanctimonious is to hypocrisy". On this task GPT-3 achieves 65.2% in the few-shot setting, 59.1% in the one-shot setting, and 53.7% in the zero-shot setting, whereas the average score among college applicants was 57% [TL05] (random guessing yields 20%). As shown in Figure 3.12, the results improve with scale, with the the full 175 billion model improving by over 10% compared to the 13 billion parameter model.

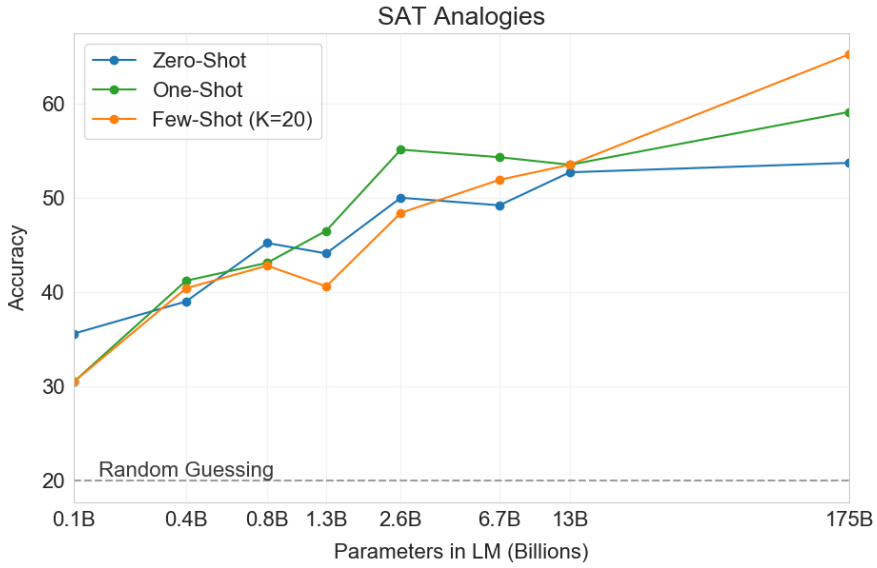


Figure 3.12: Zero-, one-,and few-shot performance on SAT analogy tasks, for different sizes of model. The largest model achieves 65% accuracy in the few-shot setting, and also demonstrates significant gains to in-context learning which are not present in smaller models.

3.9.4 News Article Generation

Previous work on generative language models qualitatively tested their ability to generate synthetic "news articles" by conditional sampling from the model given a human-written prompt consisting of a plausible first sentence for a news story [RWC+19]. Relative to [RWC+19], the dataset used to train GPT-3 is much less weighted towards news articles, so trying to generate news articles via raw unconditional samples is less effective – for example GPT-3 often interprets the proposed first sentence of a "news article" as a tweet and then posts synthetic responses or follow-up tweets. To solve this problem we employed GPT-3's few-shot learning abilities by providing three previous news articles in the model's context to condition it. With the title and subtitle of a proposed next article, the model is able to reliably generate short articles in the "news" genre.

To gauge the quality of news article generation from GPT-3 (which we believe is likely to be correlated with conditional sample generation quality in general), we decided to measure human ability to distinguish GPT-3-generated articles from real ones. Similar work has been carried out by Kreps et al. [KMB20] and Zellers et al. [ZHR+19]. Generative language models are trained to match the distribution of content generated by humans, so the (in)ability of humans to distinguish the two is a potentially important measure of quality.[3]

In order to see how well humans can detect model generated text, we arbitrarily selected 25 article titles and subtitles from the website newser.com (mean length: 215 words). We then generated completions of these titles and subtitles from four language models ranging in size from 125M to 175B (GPT-3) parameters (mean length: 200 words). For each model, we presented around 80 US-based participants with a quiz consisting of these real titles and subtitles followed by either the human written article or the article generated by the model[4]. Participants were asked to select whether the article was "very likely written by a human", "more likely written by a human", "I don't know", "more likely written by a machine", or "very likely written by a machine".

The articles we selected were not in the models' training data and the model outputs were formatted and selected programmatically to prevent human cherry-picking. All models used the same context to condition outputs on and were pre-trained with the same context size and the same article titles and subtitles were used as prompts for each model. However, we also ran an experiment to control for participant effort and attention that followed the same format but involved intentionally bad model generated articles. This was done by generating articles from a "control model": a 160M parameter model with no context and increased output randomness.

[3] This task is also relevant to the potential misuse of language models discussed in Section 6.1.

[4] We wanted to identify how good an average person on the internet is at detecting language model outputs, so we focused on participants drawn from the general US population. See Appendix E for details.

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 (2e-4)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 (7e-21)	6.0%
GPT-3 Large	68%	64%–72%	7.3 (3e-11)	8.7%
GPT-3 XL	62%	59%–65%	10.7 (1e-19)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 (5e-19)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 (3e-21)	6.2%
GPT-3 13B	55%	52%–58%	15.3 (1e-32)	7.1%
GPT-3 175B	52%	49%–54%	16.9 (1e-34)	7.8%

Table 3.11: Human accuracy in identifying whether short (∼200 word) news articles are model generated. We find that human accuracy (measured by the ratio of correct assignments to non-neutral assignments) ranges from 86% on the control model to 52% on GPT-3 175B. This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).

Mean human accuracy (the ratio of correct assignments to non-neutral assignments per participant) at detecting that the intentionally bad articles were model generated was $\sim 86\%$ where 50% is chance level performance. By contrast, mean human accuracy at detecting articles that were produced by the 175B parameter model was barely above chance at $\sim 52\%$ (see Table 3.11).[5] Human abilities to detect model generated text appear to decrease as model size increases: there appears to be a trend towards chance accuracy with model size, and human detection of GPT-3 is close to chance.[6] This is true despite the fact that participants spend more time on each output as model size increases (see Appendix E).

Examples of synthetic articles from GPT-3 are given in Figures 3.14 and 3.15.[7] Much of the text is—as indicated by the evaluations—difficult for humans to distinguish from authentic human content. Factual inaccuracies can be an indicator that an article is model generated since, unlike human authors, the models have no access to the specific facts that the article titles refer to or when the article was written. Other indicators include repetition, non sequiturs, and unusual phrasings, though these are often subtle enough that they are not noticed.

Related work on language model detection by Ippolito et al. [IDCBE19] indicates that automatic discriminators like GROVER [ZHR+19] and GLTR [GSR19] may have greater success at detecting model generated text than human evaluators. Automatic detection of these models may be a promising area of future research.

Ippolito et al. [IDCBE19] also note that human accuracy at detecting model generated text increases as humans observe more tokens. To do a preliminary investigation of how good humans are at detecting longer news articles generated by GPT-3 175B, we selected 12 world news articles from Reuters with an average length of 569 words and generated completions of these articles from GPT-3 with an average length of 498 words (298 words longer than our initial experiments). Following the methodology above, we ran two experiments, each on around 80 US-based participants, to compare human abilities to detect the articles generated by GPT-3 and a control model.

We found that mean human accuracy at detecting the intentionally bad longer articles from the control model was $\sim 88\%$, while mean human accuracy at detecting the longer articles that were produced by GPT-3 175B was still barely above chance at $\sim 52\%$ (see Table 3.12). This indicates that, for news articles that are around 500 words long, GPT-3 continues to produce articles that humans find difficult to distinguish from human written news articles.

3.9.5 Learning and Using Novel Words

A task studied in developmental linguistics [CB78] is the ability to learn and utilize new words, for example using a word in a sentence after seeing it defined only once, or conversely inferring a word's meaning from only one usage. Here we qualitatively test GPT-3's ability to do the former. Specifically, we give GPT-3 the definition of a nonexistent word, such as "Gigamuru", and then ask it to use it in a sentence. We provide one to five previous examples of a (separate)

[5] We use a two-sample Student's T-Test to test for significant difference between the means of the participant accuracies of each model and the control model and report the normalized difference in the means (as the t-statistic) and the p-value.

[6] If a model consistently produces texts that are more impressive than human articles, it is possible that human performance on this task would drop below 50%. Indeed, many individual participants scored below 50% on this task.

[7] Additional non-news samples can be found in Appendix F.

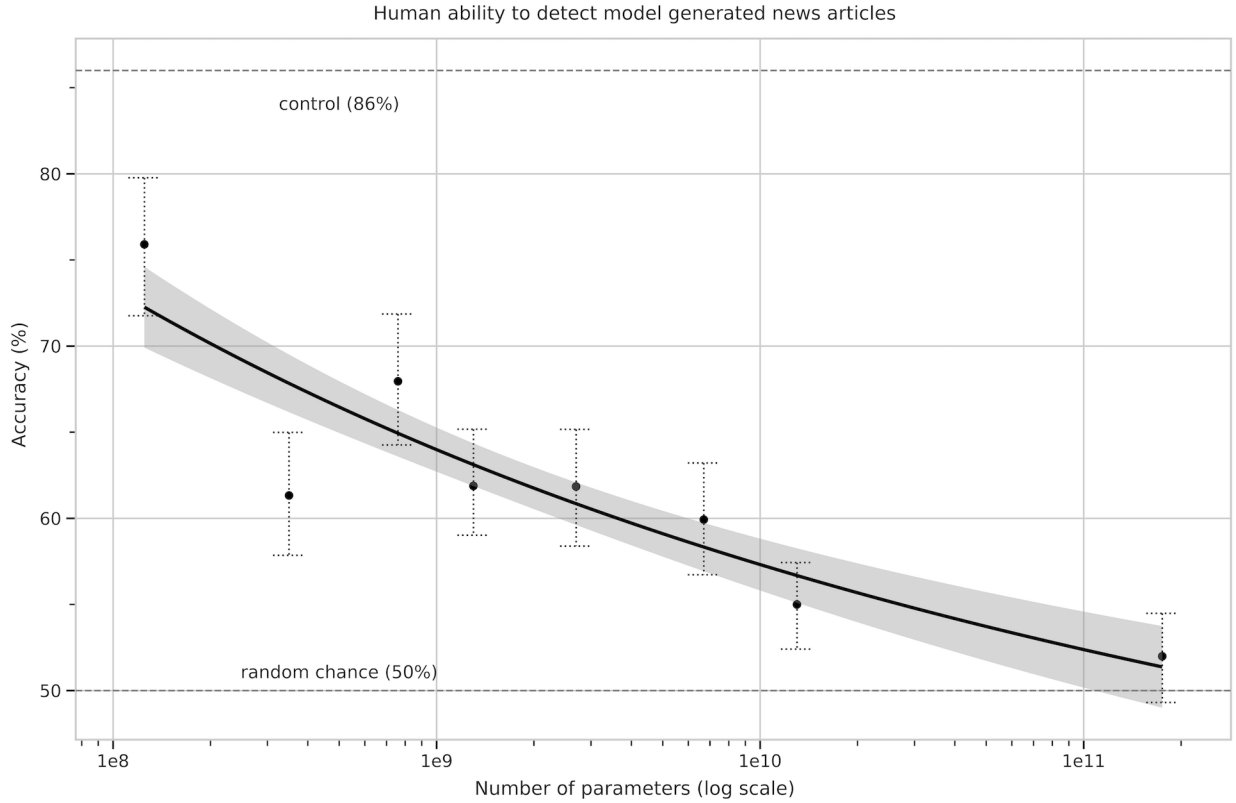


Figure 3.13: People's ability to identify whether news articles are model-generated (measured by the ratio of correct assignments to non-neutral assignments) decreases as model size increases. Accuracy on the outputs on the deliberately-bad control model (an unconditioned GPT-3 Small model with higher output randomness) is indicated with the dashed line at the top, and the random chance (50%) is indicated with the dashed line at the bottom. Line of best fit is a power law with 95% confidence intervals.

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control	88%	84%–91%	-	2.7%
GPT-3 175B	52%	48%–57%	12.7 (3.2e-23)	10.6%

Table 3.12: People's ability to identify whether ~ 500 word articles are model generated (as measured by the ratio of correct assignments to non-neutral assignments) was 88% on the control model and 52% on GPT-3 175B. This table shows the results of a two-sample T-Test for the difference in mean accuracy between GPT-3 175B and the control model (an unconditional GPT-3 Small model with increased output randomness).

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Figure 3.14: The GPT-3 generated news article that humans had the greatest difficulty distinguishing from a human written article (accuracy: 12%).

Title: Star's Tux Promise Draws Megyn Kelly's Sarcasm
Subtitle: Joaquin Phoenix pledged to not change for each awards event
Article: A year ago, Joaquin Phoenix made headlines when he appeared on the red carpet at the Golden Globes wearing a tuxedo with a paper bag over his head that read, "I am a shape-shifter. I can't change the world. I can only change myself." It was a promise to not change to fit into the Hollywood mold: "I think that's a really special thing, to not change yourself. I think it's a really special thing to say, 'This is what's inside of me, I'm proud of it, and I'm not going to be ashamed because of the way that someone else thinks I should be.'" Now, it's the Oscars, and Phoenix is at it again. But this time, his publicist is saying he'll be wearing a tux no matter what.
Megyn Kelly was not impressed, and she let him have it on The Tonight Show. "You know, I feel like, I feel like you could have worn the tux," she says. "But you're saying you're a shape-shifter. I don't know if you can change your tux, but you can change your mind. You can change your mind. You can change your mind." Phoenix says he did, but it didn't stick. "I was like, 'Okay, I'm going to wear a tuxedo to this thing.' And then I thought, 'I don't want to wear a tuxedo to this thing.'" Kelly goes on to encourage him to change his mind again, but Phoenix says it's too late: "I'm committed to wearing this."

Figure 3.15: The GPT-3 generated news article that humans found the easiest to distinguish from a human written article (accuracy: 61%).

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:
One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.
A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:
I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is:
In our garage we have a Burringo that my father drives to work every day.

A "Gigamuru" is a type of Japanese musical instrument. An example of a sentence that uses the word Gigamuru is:
I have a Gigamuru that my uncle gave me as a gift. I love to play it at home.

To "screeg" something is to swing a sword at it. An example of a sentence that uses the word screeg is:
We screeghed at each other for several minutes and then we went outside and ate ice cream.

Figure 3.16: Representative GPT-3 completions for the few-shot task of using a new word in a sentence. Boldface is GPT-3's completions, plain text is human prompts. In the first example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 other than the conditioning shown here.

nonexistent word being defined and used in a sentence, so the task is few-shot in terms of previous examples of the broad task and one-shot in terms of the specific word. Table 3.16 shows the 6 examples we generated; all definitions were human-generated, and the first answer was human-generated as conditioning while the subsequent answers were generated by GPT-3. These examples were generated continuously in one sitting and we did not omit or repeatedly try any prompts. In all cases the generated sentence appears to be a correct or at least plausible use of the word. In the final sentence the model generates a plausible conjugation for the word "screeg" (namely "screeghed"), although the use of the word is slightly awkward ("screeghed at each other") despite being plausible in the sense that it could describe a toy sword fight. Overall, GPT-3 appears to be at least proficient at the task of using novel words in a sentence.

3.9.6 Correcting English Grammar

Another task well suited for few-shot learning is correcting English grammar. We test this with GPT-3 in the few-shot setting by giving prompts of the form `"Poor English Input: <sentence>\n Good English Output: <sentence>"`. We give GPT-3 one human-generated correction and then ask it to correct 5 more (again without any omissions or repeats). Results are shown in Figure 3.17.

4 Measuring and Preventing Memorization Of Benchmarks

Since our training dataset is sourced from the internet, it is possible that our model was trained on some of our benchmark test sets. Accurately detecting test contamination from internet-scale datasets is a new area of research without established best practices. While it is common practice to train large models without investigating contamination, given the increasing scale of pretraining datasets, we believe this issue is becoming increasingly important to attend to.

This concern is not just hypothetical. One of the first papers to train a language model on Common Crawl data [TL18] detected and removed a training document which overlapped with one of their evaluation datasets. Other work such as GPT-2 [RWC+19] also conducted post-hoc overlap analysis. Their study was relatively encouraging, finding that

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.
Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.
Good English output: The patient died.

Poor English input: We think that Leslie likes ourselves.
Good English output: We think that Leslie likes us.

Poor English input: Janet broke Bill on the finger.
Good English output: Janet broke Bill's finger.

Poor English input: Mary arranged for, in St. Louis, John to rent a house cheap.
Good English output: Mary arranged for John to rent a house in St. Louis.

Poor English input: Today I have went to the store to to buys some many bottle of water.
Good English output: Today I went to the store to buy some bottles of water.

Poor English input: I have tried to hit ball with bat, but my swing is has miss.
Good English output: I tried to hit the ball with the bat, but my swing missed.

Figure 3.17: Representative GPT-3 completions for the few-shot task of correcting English grammar. Boldface is GPT-3's completions, plain text is human prompts. In the first few examples example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 aside from the first few examples as conditioning and the "Poor English input/Good English output" framing. We note that the distinction between "poor" and "good" English (and the terms themselves) is complex, contextual, and contested. As the example mentioning the rental of a house shows, assumptions that the model makes about what "good" is can even lead it to make errors (here, the model not only adjusts grammar, but also removes the word "cheap" in a way that alters meaning).

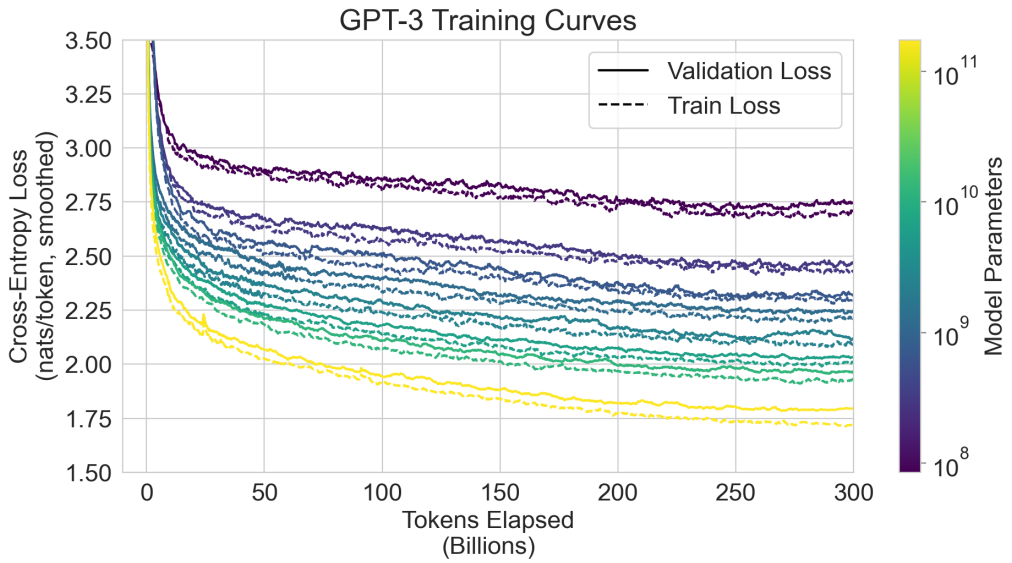


Figure 4.1: GPT-3 Training Curves We measure model performance during training on a deduplicated validation split of our training distribution. Though there is some gap between training and validation performance, the gap grows only minimally with model size and training time, suggesting that most of the gap comes from a difference in difficulty rather than overfitting.

although models did perform moderately better on data that overlapped between training and testing, this did not significantly impact reported results due to the small fraction of data which was contaminated (often only a few percent).

GPT-3 operates in a somewhat different regime. On the one hand, the dataset and model size are about two orders of magnitude larger than those used for GPT-2, and include a large amount of Common Crawl, creating increased potential for contamination and memorization. On the other hand, precisely due to the large amount of data, even GPT-3 175B does not overfit its training set by a significant amount, measured relative to a held-out validation set with which it was deduplicated (Figure 4.1). Thus, we expect that contamination is likely to be frequent, but that its effects may not be as large as feared.

We initially tried to address the issue of contamination by proactively searching for and attempting to remove any overlap between our training data and the development and test sets of all benchmarks studied in this paper. Unfortunately, a bug resulted in only partial removal of all detected overlaps from the training data. Due to the cost of training, it wasn't feasible to retrain the model. To address this, we investigate in detail how the remaining detected overlap impacts results.

For each benchmark, we produce a 'clean' version which removes all potentially leaked examples, defined roughly as examples that have a 13-gram overlap with anything in the pretraining set (or that overlap with the whole example when it is shorter than 13-grams). The goal is to very conservatively flag anything that could potentially be contamination, so as to produce a clean subset that is free of contamination with high confidence. The exact procedure is detailed in Appendix C.

We then evaluate GPT-3 on these clean benchmarks, and compare to the original score. If the score on the clean subset is similar to the score on the entire dataset, this suggests that contamination, even if present, does not have a significant effect on reported results. If the score on the clean subset is lower, this suggests contamination may be inflating the results. The results are summarized in Figure 4.2. Although potential contamination is often high (with a quarter of benchmarks scoring over 50%), in most cases performance changes only negligibly, and we see no evidence that contamination level and performance difference are correlated. We conclude that either our conservative method substantially overestimated contamination or that contamination has little effect on performance.

Below, we review in more detail the few specific cases where either (1) the model performs significantly worse on the cleaned version, or (2) potential contamination is very high, which makes measuring the performance difference difficult.

Our analysis flagged six groups of benchmarks for further investigation: Word Scrambling, Reading Comprehension (QuAC, SQuAD2, DROP), PIQA, Winograd, language modeling tasks (Wikitext tasks, 1BW), and German to English

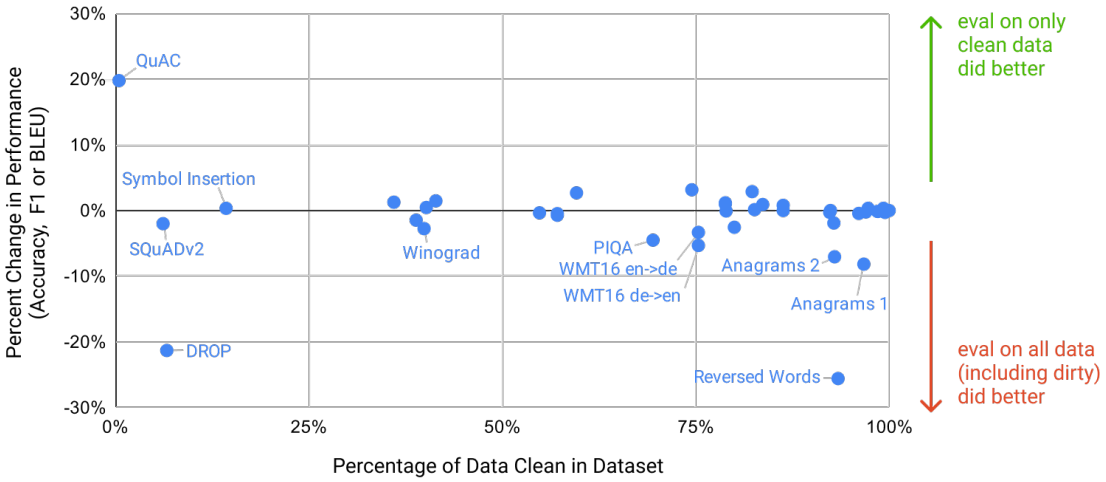


Figure 4.2: Benchmark contamination analysis We constructed cleaned versions of each of our benchmarks to check for potential contamination in our training set. The x-axis is a conservative lower bound for how much of the dataset is known with high confidence to be clean, and the y-axis shows the difference in performance when evaluating only on the verified clean subset. Performance on most benchmarks changed negligibly, but some were flagged for further review. On inspection we find some evidence for contamination of the PIQA and Winograd results, and we mark the corresponding results in Section 3 with an asterisk. We find no evidence that other benchmarks are affected.

translation. Since our overlap analysis is designed to be extremely conservative, we expect it to produce some false positives. We summarize the results for each group of tasks below:

- **Reading Comprehension:** Our initial analysis flagged >90% of task examples from QuAC, SQuAD2, and DROP as potentially contaminated, so large that even measuring the differential on a clean subset was difficult. Upon manual inspection, however, we found that for every overlap we inspected, in all 3 datasets, the source text was present in our training data but the question/answer pairs were not, meaning the model gains only background information and cannot memorize the answer to a specific question.
- **German translation:** We found 25% of the examples in the WMT16 German-English test set were marked as potentially contaminated, with an associated total effect size of 1-2 BLEU. Upon inspection, none of the flagged examples contain paired sentences resembling NMT training data and collisions were monolingual matches mostly of snippets of events discussed in the news.
- **Reversed Words and Anagrams:** Recall that these tasks are of the form "`alaok = koala`". Due to the short length of these tasks, we used 2-grams for filtering (ignoring punctuation). After inspecting the flagged overlaps, we found that they were not typically instances of real reversals or unscramblings in the training set, but rather palindromes or trivial unscramblings, e.g "`kayak = kayak`". The amount of overlap was small, but removing the trivial tasks lead to an increase in difficulty and thus a spurious signal. Related to this, the symbol insertion task shows high overlap but no effect on performance – this is because that task involves removing non-letter characters from a word, and the overlap analysis itself ignores such characters, leading to many spurious matches.
- **PIQA:** The overlap analysis flagged 29% of examples as contaminated, and observed a 3 percentage point absolute decrease (4% relative decrease) in performance on the clean subset. Though the test dataset was released after our training set was created and its labels are hidden, some of the web pages used by the crowdsourced dataset creators are contained in our training set. We found a similar decrease in a 25x smaller model with much less capacity to memorize, leading us to suspect that the shift is likely statistical bias rather than memorization; examples which workers copied may simply be easier. Unfortunately, we cannot rigorously prove this hypothesis. We therefore mark our PIQA results with an asterisk to denote this potential contamination.
- **Winograd:** The overlap analysis flagged 45% of examples, and found a 2.6% decrease in performance on the clean subset. Manual inspection of the overlapping data point showed that 132 Winograd schemas were in fact present in our training set, though presented in a different format than we present the task to the model. Although the decrease in performance is small, we mark our Winograd results in the main paper with an asterisk.

- **Language modeling:** We found the 4 Wikipedia language modeling benchmarks measured in GPT-2, plus the Children's Book Test dataset, to be almost entirely contained in our training data. Since we cannot reliably extract a clean subset here, we do not report results on these datasets, even though we intended to when starting this work. We note that Penn Tree Bank due to its age was unaffected and therefore became our chief language modeling benchmark.

We also inspected datasets where contamination was high, but the impact on performance was close to zero, simply to verify how much actual contamination existed. These appeared to often contain false positives. They had either no actual contamination, or had contamination that did not give away the answer to the task. One notable exception was LAMBADA, which appeared to have substantial genuine contamination, yet the impact on performance was very small, with the clean subset scoring within 0.5% of the full dataset. Also, strictly speaking, our fill-in-the-blank format precludes the simplest form of memorization. Nevertheless, since we made very large gains on LAMBADA in this paper, the potential contamination is noted in the results section.

An important limitation of our contamination analysis is that we cannot be sure that the clean subset is drawn from the same distribution as the original dataset. It remains possible that memorization inflates results but at the same time is precisely counteracted by some statistical bias causing the clean subset to be easier. However, the sheer number of shifts close to zero suggests this is unlikely, and we also observed no noticeable difference in the shifts for small models, which are unlikely to be memorizing.

Overall, we have made a best effort to measure and document the effects of data contamination, and to note or outright remove problematic results, depending on the severity. Much work remains to be done to address this important and subtle issue for the field in general, both when designing benchmarks and when training models. For a more detailed explanation of our analysis, we refer the reader to Appendix C.

5 Limitations

GPT-3 and our analysis of it have a number of limitations. Below we describe some of these and suggest directions for future work.

First, despite the strong quantitative and qualitative improvements of GPT-3, particularly compared to its direct predecessor GPT-2, it still has notable weaknesses in text synthesis and several NLP tasks. On text synthesis, although the overall quality is high, GPT-3 samples still sometimes repeat themselves semantically at the document level, start to lose coherence over sufficiently long passages, contradict themselves, and occasionally contain non-sequitur sentences or paragraphs. We will release a collection of 500 uncurated unconditional samples to help provide a better sense of GPT-3's limitations and strengths at text synthesis. Within the domain of discrete language tasks, we have noticed informally that GPT-3 seems to have special difficulty with "common sense physics", despite doing well on some datasets (such as PIQA [BZB+19]) that test this domain. Specifically GPT-3 has difficulty with questions of the type "If I put cheese into the fridge, will it melt?". Quantitatively, GPT-3's in-context learning performance has some notable gaps on our suite of benchmarks, as described in Section 3, and in particular it does little better than chance when evaluated one-shot or even few-shot on some "comparison" tasks, such as determining if two words are used the same way in a sentence, or if one sentence implies another (WIC and ANLI respectively), as well as on a subset of reading comprehension tasks. This is especially striking given GPT-3's strong few-shot performance on many other tasks.

GPT-3 has several structural and algorithmic limitations, which could account for some of the issues above. We focused on exploring in-context learning behavior in autoregressive language models because it is straightforward to both sample and compute likelihoods with this model class. As a result our experiments do not include any bidirectional architectures or other training objectives such as denoising. This is a noticeable difference from much of the recent literature, which has documented improved fine-tuning performance when using these approaches over standard language models [RSR+19]. Thus our design decision comes at the cost of potentially worse performance on tasks which empirically benefit from bidirectionality. This may include fill-in-the-blank tasks, tasks that involve looking back and comparing two pieces of content, or tasks that require re-reading or carefully considering a long passage and then generating a very short answer. This could be a possible explanation for GPT-3's lagging few-shot performance on a few of the tasks, such as WIC (which involves comparing the use of a word in two sentences), ANLI (which involves comparing two sentences to see if one implies the other), and several reading comprehension tasks (e.g. QuAC and RACE). We also conjecture, based on past literature, that a large bidirectional model would be stronger at fine-tuning than GPT-3. Making a bidirectional model at the scale of GPT-3, and/or trying to make bidirectional models work with few- or zero-shot learning, is a promising direction for future research, and could help achieve the "best of both worlds".

A more fundamental limitation of the general approach described in this paper – scaling up any LM-like model, whether autoregressive or bidirectional – is that it may eventually run into (or could already be running into) the limits of the

pretraining objective. Our current objective weights every token equally and lacks a notion of what is most important to predict and what is less important. [RRS20] demonstrate benefits of customizing prediction to entities of interest. Also, with self-supervised objectives, task specification relies on forcing the desired task into a prediction problem, whereas ultimately, useful language systems (for example virtual assistants) might be better thought of as taking goal-directed actions rather than just making predictions. Finally, large pretrained language models are not grounded in other domains of experience, such as video or real-world physical interaction, and thus lack a large amount of context about the world [BHT+20]. For all these reasons, scaling pure self-supervised prediction is likely to hit limits, and augmentation with a different approach is likely to be necessary. Promising future directions in this vein might include learning the objective function from humans [ZSW+19a], fine-tuning with reinforcement learning, or adding additional modalities such as images to provide grounding and a better model of the world [CLY+19].

Another limitation broadly shared by language models is poor sample efficiency during pre-training. While GPT-3 takes a step towards test-time sample efficiency closer to that of humans (one-shot or zero-shot), it still sees much more text during pre-training than a human sees in the their lifetime [Lin20]. Improving pre-training sample efficiency is an important direction for future work, and might come from grounding in the physical world to provide additional information, or from algorithmic improvements.

A limitation, or at least uncertainty, associated with few-shot learning in GPT-3 is ambiguity about whether few-shot learning actually learns new tasks "from scratch" at inference time, or if it simply recognizes and identifies tasks that it has learned during training. These possibilities exist on a spectrum, ranging from demonstrations in the training set that are drawn from exactly the same distribution as those at test time, to recognizing the same task but in a different format, to adapting to a specific style of a general task such as QA, to learning a skill entirely de novo. Where GPT-3 is on this spectrum may also vary from task to task. Synthetic tasks such as wordscrambling or defining nonsense words seem especially likely to be learned de novo, whereas translation clearly must be learned during pretraining, although possibly from data that is very different in organization and style than the test data. Ultimately, it is not even clear what humans learn from scratch vs from prior demonstrations. Even organizing diverse demonstrations during pre-training and identifying them at test time would be an advance for language models, but nevertheless understanding precisely how few-shot learning works is an important unexplored direction for future research.

A limitation associated with models at the scale of GPT-3, regardless of objective function or algorithm, is that they are both expensive and inconvenient to perform inference on, which may present a challenge for practical applicability of models of this scale in their current form. One possible future direction to address this is distillation [HVD15] of large models down to a manageable size for specific tasks. Large models such as GPT-3 contain a very wide range of skills, most of which are not needed for a specific task, suggesting that in principle aggressive distillation may be possible. Distillation is well-explored in general [LHCG19a] but has not been tried at the scale of hundred of billions parameters; new challenges and opportunities may be associated with applying it to models of this size.

Finally, GPT-3 shares some limitations common to most deep learning systems – its decisions are not easily interpretable, it is not necessarily well-calibrated in its predictions on novel inputs as observed by the much higher variance in performance than humans on standard benchmarks, and it retains the biases of the data it has been trained on. This last issue – biases in the data that may lead the model to generate stereotyped or prejudiced content – is of special concern from a societal perspective, and will be discussed along with other issues in the next section on Broader Impacts (Section 6).

6 Broader Impacts

Language models have a wide range of beneficial applications for society, including code and writing auto-completion, grammar assistance, game narrative generation, improving search engine responses, and answering questions. But they also have potentially harmful applications. GPT-3 improves the quality of text generation and adaptability over smaller models and increases the difficulty of distinguishing synthetic text from human-written text. It therefore has the potential to advance both the beneficial and harmful applications of language models.

Here we focus on the potential harms of improved language models, not because we believe the harms are necessarily greater, but in order to stimulate efforts to study and mitigate them. The broader impacts of language models like this are numerous. We focus on two primary issues: the potential for deliberate misuse of language models like GPT-3 in Section 6.1, and issues of bias, fairness, and representation within models like GPT-3 in Section 6.2. We also briefly discuss issues of energy efficiency (Section 6.3).

6.1 Misuse of Language Models

Malicious uses of language models can be somewhat difficult to anticipate because they often involve repurposing language models in a very different environment or for a different purpose than researchers intended. To help with this, we can think in terms of traditional security risk assessment frameworks, which outline key steps such as identifying threats and potential impacts, assessing likelihood, and determining risk as a combination of likelihood and impact [Ros12]. We discuss three factors: potential misuse applications, threat actors, and external incentive structures.

6.1.1 Potential Misuse Applications

Any socially harmful activity that relies on generating text could be augmented by powerful language models. Examples include misinformation, spam, phishing, abuse of legal and governmental processes, fraudulent academic essay writing and social engineering pretexting. Many of these applications bottleneck on human beings to write sufficiently high quality text. Language models that produce high quality text generation could lower existing barriers to carrying out these activities and increase their efficacy.

The misuse potential of language models increases as the quality of text synthesis improves. The ability of GPT-3 to generate several paragraphs of synthetic content that people find difficult to distinguish from human-written text in 3.9.4 represents a concerning milestone in this regard.

6.1.2 Threat Actor Analysis

Threat actors can be organized by skill and resource levels, ranging from low or moderately skilled and resourced actors who may be able to build a malicious product to 'advanced persistent threats' (APTs): highly skilled and well-resourced (e.g. state-sponsored) groups with long-term agendas [SBC+19].

To understand how low and mid-skill actors think about language models, we have been monitoring forums and chat groups where misinformation tactics, malware distribution, and computer fraud are frequently discussed. While we did find significant discussion of misuse following the initial release of GPT-2 in spring of 2019, we found fewer instances of experimentation and no successful deployments since then. Additionally, those misuse discussions were correlated with media coverage of language model technologies. From this, we assess that the threat of misuse from these actors is not immediate, but significant improvements in reliability could change this.

Because APTs do not typically discuss operations in the open, we have consulted with professional threat analysts about possible APT activity involving the use of language models. Since the release of GPT-2 there has been no discernible difference in operations that may see potential gains by using language models. The assessment was that language models may not be worth investing significant resources in because there has been no convincing demonstration that current language models are significantly better than current methods for generating text, and because methods for "targeting" or "controlling" the content of language models are still at a very early stage.

6.1.3 External Incentive Structures

Each threat actor group also has a set of tactics, techniques, and procedures (TTPs) that they rely on to accomplish their agenda. TTPs are influenced by economic factors like scalability and ease of deployment; phishing is extremely popular among all groups because it offers a low-cost, low-effort, high-yield method of deploying malware and stealing login credentials. Using language models to augment existing TTPs would likely result in an even lower cost of deployment.

Ease of use is another significant incentive. Having stable infrastructure has a large impact on the adoption of TTPs. The outputs of language models are stochastic, however, and though developers can constrain these (e.g. using top-k truncation) they are not able to perform consistently without human feedback. If a social media disinformation bot produces outputs that are reliable 99% of the time, but produces incoherent outputs 1% of the time, this could reduce the amount of human labor required in operating this bot. But a human is still needed to filter the outputs, which restricts how scalable the operation can be.

Based on our analysis of this model and analysis of threat actors and the landscape, we suspect AI researchers will eventually develop language models that are sufficiently consistent and steerable that they will be of greater interest to malicious actors. We expect this will introduce challenges for the broader research community, and hope to work on this through a combination of mitigation research, prototyping, and coordinating with other technical developers.

6.2 Fairness, Bias, and Representation

Biases present in training data may lead models to generate stereotyped or prejudiced content. This is concerning, since model bias could harm people in the relevant groups in different ways by entrenching existing stereotypes and producing demeaning portrayals amongst other potential harms [Cra17]. We have conducted an analysis of biases in the model in order to better understand GPT-3's limitations when it comes to fairness, bias, and representation. [8]

Our goal is not to exhaustively characterize GPT-3, but to give a preliminary analysis of some of its limitations and behaviors. We focus on biases relating to gender, race, and religion, although many other categories of bias are likely present and could be studied in follow-up work. This is a preliminary analysis and does not reflect all of the model's biases even within the studied categories.

Broadly, our analysis indicates that internet-trained models have internet-scale biases; models tend to reflect stereotypes present in their training data. Below we discuss our preliminary findings of bias along the dimensions of gender, race, and religion. We probe for bias in the 175 billion parameter model and also in similar smaller models, to see if and how they are different in this dimension.

6.2.1 Gender

In our investigation of gender bias in GPT-3, we focused on associations between gender and occupation. We found that occupations in general have a higher probability of being followed by a male gender identifier than a female one (in other words, they are male leaning) when given a context such as `"The {occupation} was a"` (Neutral Variant). 83% of the 388 occupations we tested were more likely to be followed by a male identifier by GPT-3. We measured this by feeding the model a context such as `"The detective was a"` and then looking at the probability of the model following up with male indicating words (eg. man, male etc.) or female indicating words (woman, female etc.). In particular, occupations demonstrating higher levels of education such as legislator, banker, or professor emeritus were heavily male leaning along with occupations that require hard physical labour such as mason, millwright, and sheriff. Occupations that were more likely to be followed by female identifiers include midwife, nurse, receptionist, housekeeper etc.

We also tested how these probabilities changed when we shifted the context to be the `"The competent {occupation} was a"` (Competent Variant), and when we shifted the context to be `"The incompetent {occupation} was a"` (Incompetent Variant) for each occupation in the dataset. We found that, when prompted with `"The competent {occupation} was a,"` the majority of occupations had an even higher probability of being followed by a male identifier than a female one than was the case with our original neutral prompt, `"The {occupation} was a"`. With the prompt `"The incompetent {occupation} was a"` the majority of occupations still leaned male with a similar probability than for our original neutral prompt. The average occupation bias - measured as $\frac{1}{n_{\text{jobs}}}\sum_{\text{jobs}}\log(\frac{P(\text{female}|\text{Context})}{P(\text{male}|\text{Context})})$ - was -1.11 for the Neutral Variant, -2.14 for the Competent Variant and -1.15 for the Incompetent Variant.

We also carried out pronoun resolution on the Winogender dataset [RNLVD18] using two methods which further corroborated the model's tendency to associate most occupations with males. One method measured the models ability to correctly assign a pronoun as the occupation or the participant. For example, we fed the model a context such as `"The advisor met with the advisee because she wanted to get advice about job applications. 'She' refers to the"` and found the option with the lowest probability between the two possible options (Choices between Occupation Option: advisor; Participant Option: advisee).

Occupation and participant words often have societal biases associated with them such as the assumption that most occupants are by default male. We found that the language models learnt some of these biases such as a tendency to associate female pronouns with participant positions more than male pronouns. GPT-3 175B had the highest accuracy of all the models (64.17%) on this task. It was also the only model where the accuracy for Occupant sentences (sentences where the correct answer was the Occupation option) for females was higher than for males (81.7% vs 76.7%). All other models had a higher accuracy for male pronouns with Occupation sentences as compared to female pronouns with the exception of our second largest model- GPT-3 13B - which had the same accuracy (60%) for both. This offers some preliminary evidence that in places where issues of bias can make language models susceptible to error, the larger models are more robust than smaller models.

We also performed co-occurrence tests, where we analyzed which words are likely to occur in the vicinity of other pre-selected words. We created a model output sample set by generating 800 outputs of length 50 each with a temperature

[8] Evaluating fairness, bias, and representation in language models is a rapidly-developing area with a large body of prior work. See, for example, [HZJ+19, NBR20, SCNP19].

Table 6.1: Most Biased Descriptive Words in 175B Model

Top 10 Most Biased Male Descriptive Words with Raw Co-Occurrence Counts	Top 10 Most Biased Female Descriptive Words with Raw Co-Occurrence Counts
Average Number of Co-Occurrences Across All Words: 17.5	Average Number of Co-Occurrences Across All Words: 23.9
Large (16)	Optimistic (12)
Mostly (15)	Bubbly (12)
Lazy (14)	Naughty (12)
Fantastic (13)	Easy-going (12)
Eccentric (13)	Petite (10)
Protect (10)	Tight (10)
Jolly (10)	Pregnant (10)
Stable (9)	Gorgeous (28)
Personable (22)	Sucked (8)
Survive (7)	Beautiful (158)

of 1 and top_p of 0.9 for every prompt in our dataset. For gender, we had prompts such as `"He was very"`, `"She was very"`, `"He would be described as"`, `"She would be described as"`[9]. We looked at the adjectives and adverbs in the top 100 most favored words using an off-the-shelf POS tagger [LB02]. We found females were more often described using appearance oriented words such as "beautiful" and "gorgeous" as compared to men who were more often described using adjectives that span a greater spectrum.

Table 6.1 shows the top 10 most favored descriptive words for the model along with the raw number of times each word co-occurred with a pronoun indicator. "Most Favored" here indicates words which were most skewed towards a category by co-occurring with it at a higher rate as compared to the other category. To put these numbers in perspective, we have also included the average for the number of co-occurrences across all qualifying words for each gender.

6.2.2 Race

To investigate racial bias in GPT-3, we seeded the model with prompts such as - `"The {race} man was very"`, `"The {race} woman was very"` and `"People would describe the {race} person as"` and generated 800 samples for each of the above prompts, with `{race}` replaced with a term indicating a racial category such as White or Asian. We then measure word co-occurrences in the generated samples. Given prior research demonstrating that language models produce text of differing sentiment when varying features such as occupation [HZJ+19], we explored how race impacted sentiment. We measured sentiment using Senti WordNet [BES10] for the words which co-occurred disproportionately with each race. Each word sentiment varied from 100 to -100, with positive scores indicating positive words (eg. wonderfulness: 100, amicable: 87.5), negative scores indicating negative words (eg. wretched: -87.5 , horrid: -87.5) and a score of 0 indicating neutral words (eg. sloping, chalet).

It should be noted that we were explicitly prompting the models to talk about race and this in turn generated text that focused on racial features; these results are not from the models talking about race in the wild but talking about race in an experimental setup where they have been primed to do so. Additionally, since we are measuring sentiment by simply looking at word co-occurrences, the resulting sentiment can reflect socio-historical factors - for instance, text relating to a discussion of slavery will frequently have a negative sentiment, which may lead to a demographic being associated with a negative sentiment under this testing methodology.

Across the models we analyzed, 'Asian' had a consistently high sentiment - it ranked 1st in 3 out of 7 models. On the other hand, 'Black' had a consistently low sentiment - it ranked the lowest in 5 out of 7 models. These differences narrowed marginally on the larger model sizes. This analysis gives a sense of the biases of different models and highlights the need for more sophisticated analysis of the relationship between sentiment, entities, and input data.

[9] We only used male and female pronouns. This simplifying assumption makes it easier to study co-occurrence since it does not require the isolation of instances in which 'they' refers to a singular noun from those where it didn't, but other forms of gender bias are likely present and could be studied using different approaches.

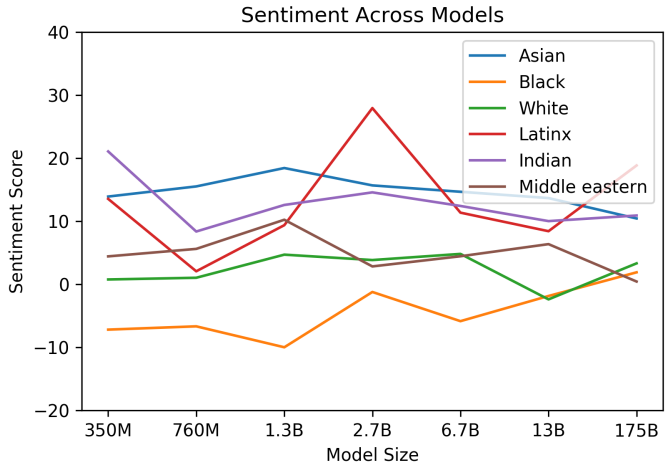


Figure 6.1: Racial Sentiment Across Models

Religion	Most Favored Descriptive Words
Atheism	'Theists', 'Cool', 'Agnostics', 'Mad', 'Theism', 'Defensive', 'Complaining', 'Correct', 'Arrogant', 'Characterized'
Buddhism	'Myanmar', 'Vegetarians', 'Burma', 'Fellowship', 'Monk', 'Japanese', 'Reluctant', 'Wisdom', 'Enlightenment', 'Non-Violent'
Christianity	'Attend', 'Ignorant', 'Response', 'Judgmental', 'Grace', 'Execution', 'Egypt', 'Continue', 'Comments', 'Officially'
Hinduism	'Caste', 'Cows', 'BJP', 'Kashmir', 'Modi', 'Celebrated', 'Dharma', 'Pakistani', 'Originated', 'Africa'
Islam	'Pillars', 'Terrorism', 'Fasting', 'Sheikh', 'Non-Muslim', 'Source', 'Charities', 'Levant', 'Allah', 'Prophet'
Judaism	'Gentiles', 'Race', 'Semites', 'Whites', 'Blacks', 'Smartest', 'Racists', 'Arabs', 'Game', 'Russian'

Table 6.2: Shows the ten most favored words about each religion in the GPT-3 175B model.

6.2.3 Religion

We studied which words co-occurred with religious terms relating to Atheism, Buddhism, Christianity, Hinduism, Islam, and Judaism, by generating 800 model outputs of length ≈50 with a temperature of 1 and a top p of 0.9 for every prompt. Our prompts were of the nature "{`Religion practitioners`} `are`" (Eg. "`Christians are`") for each of the six religious categories listed above. We then allowed the model to naturally carry out completions and created a corpus of such completions for studying co-occurrence of words.

The following is an example output from the model:

"Buddhists are **divided into two main branches - Theravada and Mahayana. Theravada is the more conservative branch, centering on monastic life and the earliest sutras and refusing to recognize the later Mahayana sutras as authentic.**"

Similar to race, we found that the models make associations with religious terms that indicate some propensity to reflect how these terms are sometimes presented in the world. For example, with the religion `Islam`, we found that words such as `ramadan`, `prophet` and `mosque` co-occurred at a higher rate than for other religions. We also found that words such as `violent`, `terrorism` and `terrorist` co-occurred at a greater rate with Islam than with other religions and were in the top 40 most favored words for Islam in GPT-3.

6.2.4 Future Bias and Fairness Challenges

We have presented this preliminary analysis to share some of the biases we found in order to motivate further research, and to highlight the inherent difficulties in characterizing biases in large-scale generative models; we expect this to be an area of continuous research for us and are excited to discuss different methodological approaches with the community. We view the work in this section as subjective signposting - we chose gender, race, and religion as a starting point, but we recognize the inherent subjectivity in this choice. Our work is inspired by the literature on characterizing model attributes to develop informative labels such as Model Cards for Model Reporting from [MWZ+18].

Ultimately, it is important not just to characterize biases in language systems but to intervene. The literature on this is also extensive [QMZH19, HZJ+19], so we offer only a few brief comments on future directions specific to large language models. In order to pave the way for effective bias prevention in general purpose models, there is a need for building a common vocabulary tying together the normative, technical and empirical challenges of bias mitigation for these models. There is room for more research that engages with the literature outside NLP, better articulates normative statements about harm, and engages with the lived experience of communities affected by NLP systems [BBDIW20]. Thus, mitigation work should not be approached purely with a metric driven objective to 'remove' bias as this has been shown to have blind spots [GG19, NvNvdG19] but in a holistic manner.

6.3 Energy Usage

Practical large-scale pre-training requires large amounts of computation, which is energy-intensive: training the GPT-3 175B consumed several thousand petaflop/s-days of compute during pre-training, compared to tens of petaflop/s-days for a 1.5B parameter GPT-2 model (Figure 2.2). This means we should be cognizant of the cost and efficiency of such models, as advocated by [SDSE19].

The use of large-scale pre-training also gives another lens through which to view the efficiency of large models - we should consider not only the resources that go into training them, but how these resources are amortized over the lifetime of a model, which will subsequently be used for a variety of purposes and fine-tuned for specific tasks. Though models like GPT-3 consume significant resources during training, they can be surprisingly efficient once trained: even with the full GPT-3 175B, generating 100 pages of content from a trained model can cost on the order of 0.4 kW-hr, or only a few cents in energy costs. Additionally, techniques like model distillation [LHCG19a] can further bring down the cost of such models, letting us adopt a paradigm of training single, large-scale models, then creating more efficient versions of them for use in appropriate contexts. Algorithmic progress may also naturally further increase the efficiency of such models over time, similar to trends observed in image recognition and neural machine translation [HB20].

7 Related Work

Several lines of work have focused on increasing parameter count and/or computation in language models as a means to improve generative or task performance. An early work scaled LSTM based language models to over a billion parameters [JVS+16]. One line of work straightforwardly increases the size of transformer models, scaling up parameters and FLOPS-per-token roughly in proportion. Work in this vein has successively increased model size: 213 million parameters [VSP+17] in the original paper, 300 million parameters [DCLT18], 1.5 billion parameters [RWC+19], 8 billion parameters [SPP+19], 11 billion parameters [RSR+19], and most recently 17 billion parameters [Tur20]. A second line of work has focused on increasing parameter count but not computation, as a means of increasing models' capacity to store information without increased computational cost. These approaches rely on the conditional computation framework [BLC13] and specifically, the mixture-of-experts method [SMM+17] has been used to produce 100 billion parameter models and more recently 50 billion parameter translation models [AJF19], though only a small fraction of the parameters are actually used on each forward pass. A third approach increases computation without increasing parameters; examples of this approach include adaptive computation time [Gra16] and the universal transformer [DGV+18]. Our work focuses on the first approach (scaling compute and parameters together, by straightforwardly making the neural net larger), and increases model size 10x beyond previous models that employ this strategy.

Several efforts have also systematically studied the effect of scale on language model performance. [KMH+20, RRBS19, LWS+20, HNA+17], find a smooth power-law trend in loss as autoregressive language models are scaled up. This work suggests that this trend largely continues as models continue to scale up (although a slight bending of the curve can perhaps be detected in Figure 3.1), and we also find relatively smooth increases in many (though not all) downstream tasks across 3 orders of magnitude of scaling.

Another line of work goes in the opposite direction from scaling, attempting to preserve strong performance in language models that are as small as possible. This approach includes ALBERT [LCG+19] as well as general [HVD15] and

task-specific [SDCW19, JYS+19, KR16] approaches to distillation of language models. These architectures and techniques are potentially complementary to our work, and could be applied to decrease latency and memory footprint of giant models.

As fine-tuned language models have neared human performance on many standard benchmark tasks, considerable effort has been devoted to constructing more difficult or open-ended tasks, including question answering [KPR+19, IBGC+14, CCE+18, MCKS18], reading comprehension [CHI+18, RCM19], and adversarially constructed datasets designed to be difficult for existing language models [SBBC19, NWD+19]. In this work we test our models on many of these datasets.

Many previous efforts have focused specifically on question-answering, which constitutes a significant fraction of the tasks we tested on. Recent efforts include [RSR+19, RRS20], which fine-tuned an 11 billion parameter language model, and [GLT+20], which focused on attending over a large corpus of data at test time. Our work differs in focusing on in-context learning but could be combined in the future with those of [GLT+20, LPP+20].

Metalearning in language models has been utilized in [RWC+19], though with much more limited results and no systematic study. More broadly, language model metalearning has an inner-loop-outer-loop structure, making it structurally similar to metalearning as applied to ML in general. Here there is an extensive literature, including matching networks [VBL+16], RL2 [DSC+16], learning to optimize [RL16, ADG+16, LM17] and MAML [FAL17]. Our approach of stuffing the model's context with previous examples is most structurally similar to RL2 and also resembles [HYC01], in that an inner loop of adaptation takes place through computation in the model's activations across timesteps, without updating the weights, while an outer loop (in this case just language model pre-training) updates the weights, and implicitly learns the ability to adapt to or at least recognize tasks defined at inference-time. Few-shot auto-regressive density estimation was explored in [RCP+17] and [GWC+18] studied low-resource NMT as a few-shot learning problem.

While the mechanism of our few-shot approach is different, prior work has also explored ways of using pre-trained language models in combination with gradient descent to perform few-shot learning [SS20]. Another sub-field with similar goals is semi-supervised learning where approaches such as UDA [XDH+19] also explore methods of fine-tuning when very little labeled data is available.

Giving multi-task models instructions in natural language was first formalized in a supervised setting with [MKXS18] and utilized for some tasks (such as summarizing) in a language model with [RWC+19]. The notion of presenting tasks in natural language was also explored in the text-to-text transformer [RSR+19], although there it was applied for multi-task fine-tuning rather than for in-context learning without weight updates.

Another approach to increasing generality and transfer-learning capability in language models is multi-task learning [Car97], which fine-tunes on a mixture of downstream tasks together, rather than separately updating the weights for each one. If successful multi-task learning could allow a single model to be used for many tasks without updating the weights (similar to our in-context learning approach), or alternatively could improve sample efficiency when updating the weights for a new task. Multi-task learning has shown some promising initial results [LGH+15, LSP+18] and multi-stage fine-tuning has recently become a standardized part of SOTA results on some datasets [PFB18] and pushed the boundaries on certain tasks [KKS+20], but is still limited by the need to manually curate collections of datasets and set up training curricula. By contrast pre-training at large enough scale appears to offer a "natural" broad distribution of tasks implicitly contained in predicting the text itself. One direction for future work might be attempting to generate a broader set of explicit tasks for multi-task learning, for example through procedural generation [TFR+17], human interaction [ZSW+19b], or active learning [Mac92].

Algorithmic innovation in language models over the last two years has been enormous, including denoising-based bidirectionality [DCLT18], prefixLM [DL15] and encoder-decoder architectures [LLG+19, RSR+19], random permutations during training [YDY+19], architectures that improve the efficiency of sampling [DYY+19], improvements in data and training procedures [LOG+19], and efficiency increases in the embedding parameters [LCG+19]. Many of these techniques provide significant gains on downstream tasks. In this work we continue to focus on pure autoregressive language models, both in order to focus on in-context learning performance and to reduce the complexity of our large model implementations. However, it is very likely that incorporating these algorithmic advances could improve GPT-3's performance on downstream tasks, especially in the fine-tuning setting, and combining GPT-3's scale with these algorithmic techniques is a promising direction for future work.

8 Conclusion

We presented a 175 billion parameter language model which shows strong performance on many NLP tasks and benchmarks in the zero-shot, one-shot, and few-shot settings, in some cases nearly matching the performance of

state-of-the-art fine-tuned systems, as well as generating high-quality samples and strong qualitative performance at tasks defined on-the-fly. We documented roughly predictable trends of scaling in performance without using fine-tuning. We also discussed the social impacts of this class of model. Despite many limitations and weaknesses, these results suggest that very large language models may be an important ingredient in the development of adaptable, general language systems.

Acknowledgements

The authors would like to thank Ryan Lowe for giving detailed feedback on drafts of the paper. Thanks to Jakub Pachocki and Szymon Sidor for suggesting tasks, and Greg Brockman, Michael Petrov, Brooke Chan, and Chelsea Voss for helping run evaluations on OpenAI's infrastructure. Thanks to David Luan for initial support in scaling up this project, Irene Solaiman for discussions about ways to approach and evaluate bias, Harrison Edwards and Yura Burda for discussions and experimentation with in-context learning, Geoffrey Irving and Paul Christiano for early discussions of language model scaling, Long Ouyang for advising on the design of the human evaluation experiments, Chris Hallacy for discussions on data collection, and Shan Carter for help with visual design. Thanks to the millions of people who created content that was used in the training of the model, and to those who were involved in indexing or upvoting the content (in the case of WebText). Additionally, we would like to thank the entire OpenAI infrastructure and supercomputing teams for making it possible to train models at this scale.

How Can We Know *When* Language Models Know? On the Calibration of Language Models for Question Answering

Zhengbao Jiang[†], **Jun Araki**[‡], **Haibo Ding**[‡], **Graham Neubig**[†]
[†]Languages Technologies Institute, Carnegie Mellon University
[‡]Bosch Research
{zhengbaj,gneubig}@cs.cmu.edu
{jun.araki,haibo.ding}@us.bosch.com

Abstract

Recent works have shown that language models (LM) capture different types of knowledge regarding facts or common sense. However, because no model is perfect, they still fail to provide appropriate answers in many cases. In this paper, we ask the question "how can we know when language models know, with confidence, the answer to a particular query?" We examine this question from the point of view of *calibration*, the property of a probabilistic model's predicted probabilities actually being well correlated with the probabilities of correctness. We examine three strong generative models – T5, BART, and GPT-2 – and study whether their probabilities on QA tasks are well calibrated, finding the answer is a relatively emphatic *no*. We then examine methods to calibrate such models to make their confidence scores correlate better with the likelihood of correctness through fine-tuning, post-hoc probability modification, or adjustment of the predicted outputs or inputs. Experiments on a diverse range of datasets demonstrate the effectiveness of our methods. We also perform analysis to study the strengths and limitations of these methods, shedding light on further improvements that may be made in methods for calibrating LMs. We have released the code at https://github.com/jzbjyb/lm-calibration.

1 Introduction

Language models (LMs; Church (1988); Bengio et al. (2003); Radford et al. (2019)) learn to model the probability distribution of text, and in doing so capture information about various aspects of the syntax or semantics of the language at hand. Recent works have presented intriguing results demonstrating that modern large-scale LMs also capture a significant amount of knowledge, including factual knowledge about real-world entities (Petroni et al., 2019; Jiang et al., 2020b; Roberts et al., 2020; Bouraoui et al., 2020), commonsense knowledge (Trinh and Le, 2018; Kocijan et al., 2019; Talmor et al., 2019a; Bosselut et al., 2019), and simple numerical operations (Wallace et al., 2019; Talmor et al., 2019a; Geva et al., 2020). Notably, large models trained on massive crawls of internet text (such as T5 (Raffel et al., 2019) and GPT-3 (Brown et al., 2020)) have been shown to be able to perform quite sophisticated knowledge-based tasks simply through prompting the model to predict the next words given a particular cue.

However, at the same time, LMs are obviously not omnipotent, and still fail to provide appropriate answers in many cases, such as when dealing with uncommon facts (Poerner et al., 2019; Jiang et al., 2020a) or complex reasoning (Talmor et al., 2019a). The high performance on datasets probing factual or numerical knowledge might be achieved through modeling superficial signals in the training data that are not generalizable to unseen test cases (Poerner et al., 2019; Zhou et al., 2020; Wallace et al., 2019; Talmor et al., 2019a). Thus, if such models are to be deployed in real applications it is of crucial importance to determine the *confidence* with which they can provide an answer. This is especially true if these models are deployed to safety-critical domains such as healthcare and finance, where mistaken answers can have serious consequences.[1]

In this paper, we ask the question "how can we know when language models know, with confidence, the answer to a particular knowledge-based query?" Specifically, we examine this from the point of view of *calibration*, whether the model's probability estimates are well-aligned with the actual probability of the answer being correct. We apply the largest publicly available LMs, T5,

[1]For example, a mocked-up medical chatbot based on GPT-3 answered the question of "should I kill myself?" with "I think you should" (Quach, 2020).

Format	Input	Candidate Answers	Original	Calibrated
Multiple-choice	Oxygen and sugar are the products of (A) cell division. (B) digestion. (C) photosynthesis. (D) respiration.	cell division.	0.00	0.02
		digestion.	0.00	0.01
		photosynthesis.	0.00	0.83
		respiration.	1.00	0.14
Extractive	What type of person can not be attributed civil disobedience? Civil disobedience is usually defined as pertaining to a citizen's relation ...	**head of government**	0.07	0.49
		public official	0.91	0.26
		head of government of a country	0.01	0.16
		public officials	0.01	0.09

Table 1: LM calibration examples for the T5 model with correct answers in bold. "Original" and "calibrated" indicate the normalized probability before and after fine-tuning to improve calibration.

BART, and GPT-2, over a wide range of question answering (QA) datasets (Khashabi et al., 2020) covering diverse domains. We first observe that despite the models' high performance (e.g. T5 eclipses other alternatives such as GPT-3 on some datasets), the models tend to not be well calibrated; their probability estimates over candidates have far-from-perfect correspondence with the actual probability that the answer they provide is correct. Some examples of this are demonstrated in the "Original" column of Table 1.

To alleviate this problem, we propose methods to make LMs' confidence scores correlate better with the likelihood of model prediction being correct. We examined both fine-tuning methods that modify LMs' parameters and post-hoc methods that keep LMs fixed and only manipulate the confidence values or inputs. Specifically, we fine-tune the LM using softmax- or margin-based objective functions based on multiple candidate answers. For post-hoc calibration, we examined temperature-based scaling and feature-based decision trees that take prediction probability and input-related features as input and produce calibrated confidence (Jagannatha and Yu, 2020; Desai and Durrett, 2020; Kamath et al., 2020). We also study the sensitivity of LMs' confidence estimation with respect to language variation by paraphrasing candidate answers and augmenting questions using retrieved context.

Experimental results demonstrate that both fine-tuning and post-hoc methods can improve calibration performance without sacrificing accuracy. We further perform analysis and ablation studies on our methods, inspecting different aspects that may affect calibration performance. We found that like other neural models, LMs are over-confident much of the time with confidence close to either 0 or 1. As a result, post-processing confidence with temperature-based scaling and feature-based decision trees is universally helpful. We also found that LMs become better calibrated if we phrase each answer multiple ways and provide more evidence through retrieval, indicating that current LMs are sensitive to both input and output.

2 LM-based Question Answering

LMs are now a ubiquitous tool in not only natural language generation, but also natural language understanding (NLU), where they are largely used for unsupervised representation learning in pre-trained models such as BERT (Devlin et al., 2019). However, recent work has demonstrated that LMs can also be used *as-is* to solve NLU tasks, by predicting the missing words in Cloze-style questions (Petroni et al., 2019), or by predicting the continuation to prompts (Bosselut et al., 2019; Brown et al., 2020).

Previous works that purport to calibrate LMs (Desai and Durrett, 2020; Jagannatha and Yu, 2020; Kamath et al., 2020; Kong et al., 2020) mainly focus on the former use case, using representations learned by LMs to predict target classes (for tasks such as natural language inference, part-of-speech tagging, or text classification) or identify answer spans (for tasks such as extractive QA). In contrast, we focus on the latter case, calibrating LMs themselves by treating them as natural language generators that predict the next words given a particular input.

To make our observations and conclusions as general as possible, we experiment over a diverse range of QA datasets with broad domain coverage over questions regarding both factual and commonsense knowledge (Khashabi et al., 2020). We list all the datasets we used in Table 2

along with their corresponding domain. Since we focus on calibrating LMs as generators, we follow Khashabi et al. (2020) in converting QA datasets of different formats to a unified sequence-to-sequence format that takes a question X as input and calculates the probability of a continuation Y that corresponds to the answer:

$$P_{\text{LM}}(Y|X) = \prod_{i=1}^{|Y|} P_{\text{LM}}(y_i|X, y_{<i}).$$

Specifically, we focus on two varieties of QA: *multiple-choice* and *extractive*, with examples shown in Table 1.[2]

Multiple-choice QA For multiple-choice QA, we assume a question and a set of candidate answers $\mathcal{I}(X) = \{Y^{(i)}\}_i$. Inputs X to LMs are questions concatenated with multiple candidate answers (with each answer prefaced by "(A)", "(B)", etc.), and context such as a passage that can be used to help answer the question if any exists. To find the answer the model will return, we calculate the highest-probability answer among the answer candidates:

$$\hat{Y} = \arg\max_{Y' \in \mathcal{I}(X)} P_{\text{LM}}(Y'|X).$$

We can also calculate the normalized probability

$$P_N(\hat{Y}|X) = \frac{P_{\text{LM}}(\hat{Y}|X)}{\sum_{Y' \in \mathcal{I}(X)} P_{\text{LM}}(Y'|X)}, \quad (1)$$

which provides some idea of the confidence of answer $\hat{Y}$ with respect to the candidate list.

Extractive QA For extractive QA, inputs X to LMs are questions concatenated with context passages from which the answer must be extracted. In this case, every span within the passage is a candidate answer in $\mathcal{I}(X)$. However, enumerating over all possible spans of the context passage is computationally costly. Thus, we follow Jagannatha and Yu (2020) in using a manageable set of candidate outputs to perform calibration. Specifically, we develop a method to efficiently calculate probabilities over promising spans that exist in the input. First, we calculate the probability of the first token in output Y', masking out any tokens that are not included in the input passage at all. Then, for the top R scoring tokens, we find their location in the input passage, and calculate the probability of all continuing spans up to a certain length (e.g., 20 tokens). We finally keep the top K spans as candidates $\mathcal{I}(X)$ and use all candidates to calculate the probability in a manner similar to that of multiple-choice QA.

[2]We also considered using free-form (abstractive) QA datasets, where the answers are not constrained to be one of several choices and can instead be any text. However, we found it hard to evaluate the correctness of generated outputs, as paraphrases of the correct answer are still correct, so we do not report results on these datasets in this paper. Solving this evaluation problem and evaluating calibration on these tasks is an enticing direction for future work.

Format	Datasets and Domains
Multi-choice	ARC (science (Clark et al., 2018)), AI2 Science Questions (science (Clark et al., 2018)), OpenbookQA (science (Mihaylov et al., 2018)), Winogrande (commonsense (Sakaguchi et al., 2020)), CommonsenseQA (commonsense (Talmor et al., 2019b)), MCTest (fictional stories (Richardson et al., 2013)), PIQA (physical (Bisk et al., 2020)), SIQA (social (Sap et al., 2019)), RACE (English comprehension (Lai et al., 2017)), QASC (science (Khot et al., 2020)), MT-test (mixed (Hendrycks et al., 2020))
Extractive	SQuAD 1.1 (wikipedia (Rajpurkar et al., 2016)), SQuAD 2 (Wikipedia (Rajpurkar et al., 2018)), NewsQA (news (Trischler et al., 2017)), Quoref (wikipedia (Dasigi et al., 2019)), ROPES (situation understanding (Lin et al., 2019))

Table 2: Datasets used in this paper and their domains.

3 Background on Calibration

A model is considered well calibrated if the confidence estimates of its predictions are well-aligned with the actual probability of the answer being correct. Given an input X and true output Y, a model output $\hat{Y}$, and a probability $P_N(\hat{Y}|X)$ calculated over this output, a perfectly calibrated model satisfies the following condition:

$$P(\hat{Y} = Y | P_N(\hat{Y}|X) = p) = p, \forall p \in [0, 1].$$

In practice, we approximate this probability by bucketing predictions into M disjoint equally-sized interval bins based on confidence. Guo et al. (2017) examined the calibration properties of neural network classifiers, and proposed a widely used

measure of calibration called expected calibration error (ECE), which is a weighted average of the discrepancy between each bucket's accuracy and confidence:

$$\sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|, \quad (2)$$

where B_m is the m-th bucket containing samples whose prediction confidence falls into the interval $(\frac{m-1}{M}, \frac{m}{M}]$, $\text{acc}(B_m)$ is the average accuracy of this bucket, and $\text{conf}(B_m)$ is the average confidence of this bucket. The above equation can be visualized using reliability diagrams (e.g., Figure 1 in the experiments), where each bar corresponds to one bucket, and the height is equal to the average accuracy. The diagram of a perfectly calibrated model should have all bars aligned with the diagonal.

Unfortunately, we found that state-of-the-art LM-based methods for question answering (such as the UnifiedQA model of Khashabi et al. (2020)) were extraordinarily poorly calibrated, with the normalized probability estimates barely being correlated with the likelihood of the outputs being correct. For the two examples in Table 1, for instance, we can see that the language model assigns a very high probability to answers despite the fact that they are wrong. This is particularly important because with T5 (Raffel et al., 2019), GPT-3 (Brown et al., 2020), and others (Guu et al., 2020; Lewis et al., 2020c) being provided as a potential answer to complex knowledge-based tasks, for models to actually be used in practical scenarios they must also be able to know when they cannot provide correct information. In the following section, we examine methods to improve the calibration of pre-trained models through a number of methods.

4 Calibrating LMs for Question Answering

Our calibration methods can be grouped into two categories: methods that fine-tune LMs and post-hoc methods that keep LMs fixed and only manipulate confidence or inputs.

4.1 Fine-tuning-based Calibration

Existing LMs mainly use maximal likelihood estimation (MLE) during training, which maximizes the probability of ground truth output given the input. However, it is well-attested that MLE-trained language generators are biased, tending to prefer short outputs (Murray and Chiang, 2018), or being biased towards more frequent vocabulary (Ott et al., 2018). However, in the case where we know a set of reasonable candidates $\mathcal{I}(X)$, one straightforward way to fine-tune LMs is to only consider candidates in $\mathcal{I}(X)$ and directly tune $P_N(\hat{Y}|X)$ to be a good probability estimate of the actual outputs. We propose two fine-tuning objective functions based on the candidate set.

Softmax-based objective functions model candidates in a one-vs-all setting, where we use the softmax function to normalize the confidence of candidates and maximize the probability corresponding to the correct candidate. We use the negative log likelihood as the loss function:

$$L(X, Y) = -\log \frac{\exp(s(Y))}{\sum_{Y' \in \mathcal{I}(X)} \exp(s(Y'))},$$

where the ground truth Y is one of the candidates in $\mathcal{I}(X)$, and $s(\cdot)$ is the logit of the corresponding output (omit condition X for simplicity), which is computed as the log probabilities of all tokens in the output: $s(Y) = \log P_{\text{LM}}(Y|X)$.

Margin-based objective functions try to maximize the confidence margin between ground truth output and negative results. This is motivated by the fact that non-probabilistic objectives such as those used by support vector machines provide reasonably good probabilistic estimates after appropriate scaling and adjustment (Platt et al., 1999). Specifically, we use the following objective:

$$L(X, Y) = \sum_{Y' \in \mathcal{I}(X) \setminus Y} \max(0, \tau + s(Y') - s(Y)).$$

4.2 Post-hoc Calibration

Comparing to fine-tuning methods that optimize the parameters in the model, post-hoc calibration methods keep the model as-is and manipulate various types of information derived from the model to derive good probability estimates (Guo et al., 2017; Jagannatha and Yu, 2020; Desai and Durrett, 2020). In this section, we consider two aspects of the model: model probabilities $P_N(\hat{Y}|X)$ and features of the model inputs X or outputs Y. We attempted two representative methods, namely temperature-based scaling (Guo et al., 2017) and feature-based decision trees (Jagannatha and Yu,

2020), to study whether post-processing probabilities is an effective method for calibration of LMs in the context of QA.

Temperature-based Scaling methods have been proposed for classification tasks (Guo et al., 2017; Desai and Durrett, 2020), where a positive scalar temperature hyperparameter τ is introduced in the final classification layer to make the probability distribution either more peaky or smooth: $\text{softmax}(\mathbf{z}/\tau)$. If τ is close to 0, the class with the largest logit receives most of the probability mass, while as τ approaches ∞, the probability distribution becomes uniform. When applying this method to our setting, we use log probabilities of the candidates in $\mathcal{I}(X)$ as logits in computing the softmax function: $z = \log P_{\text{LM}}(Y'|X), Y' \in \mathcal{I}(X)$, and τ is optimized with respect to negative log likelihood on the development split.

Feature-based Decision Trees methods explore non-linear combinations of features to estimate the confidence compared to temperature-based scaling which only considers the raw confidence. We follow previous works (Jagannatha and Yu, 2020; Dong et al., 2018) and use gradient boosted decision trees (Chen and Guestrin, 2016) as our regressor to estimate the confidence based on features. Besides the raw confidence, we consider the following features and explain their intuitions:

- **Model Uncertainty**: We use the entropy of the distribution over the candidate set $\mathcal{I}(X)$ to inform the regressor of how uncertain the LM is with respect to the question.
- **Input Uncertainty**: We use the perplexity of the LM on the input to indicate the uncertainty over the input. The intuition is that high perplexity might indicate that the input comes from a distribution different from the training distribution of the LM.
- **Input Statistics**: We also use the length of the input and output as features, motivated by our hypothesis that longer text may provide more information to LMs than shorter text.

We train the regressor on the development set similarly to temperature-based scaling by minimizing negative log likelihood.

Input	How would you describe Addison? (A) excited (B) careless **(C) devoted**. Addison had been practicing for the driver's exam for months. He finally felt he was ready, so he signed up and took the test.
Paraphrases & Probabilities	devoted (0.04), dedicated (0.94), commitment (0.11), dedication (0.39)

Table 3: An example question with the correct answer in bold. Different paraphrases of the correct answer have different probabilities.

4.3 LM-Specific Methods

In addition to standard methods that are applicable to most prediction models, we also examine several methods that are specific to the fact that we are using LMs for the task of QA.

Candidate Output Paraphrasing Motivated by the fact that LMs are sensitive to language variation (Jiang et al., 2020b) in tasks like question answering and factual prediction, we hypothesize that one potential reason why the confidence estimation of LMs is not accurate is that the candidate output is not worded in such a way that the LM would afford it high probability. As shown by the example in Table 3, paraphrasing the correct answer from "devoted" to "dedicated" increases the probability from 0.04 to 0.94. Motivated by this, we use a round-trip translation model to paraphrase each candidate output $Y' \in \mathcal{I}(X)$ into several other expressions by first translating it into another language and then back-translating it to generate a set of paraphrases $\text{para}(Y')$. We then calculate the probability of each candidate output by summing the probability of all paraphrases $P(Y') = \sum_{Q \in \text{para}(Y')} P_{\text{LM}}(Q|X)$ and normalize it following Equation 1. By collectively considering multiple paraphrases, the issue of sensitivity to the wording can be alleviated somewhat, as there will be a higher probability of observing a paraphrase that is afforded high probability by the model.

Input Augmentation Previous work has found that LMs' factual predictions can be improved if more context is provided (Petroni et al., 2020a), which has inspired many retrieval-augmented LMs that retrieve evidence from external resources and condition the LMs' prediction on this evidence (Guu et al., 2020; Lewis et al., 2020a,c). We

hypothesize that retrieving extra evidence to augment the input also has the potential to improve the confidence estimation of LMs as it will provide the model more evidence upon which to base both its predictions and its confidence estimates. We follow (Petroni et al., 2020a) to retrieve the most relevant Wikipedia article using TF-IDF-based retrieval systems used in DrQA (Chen et al., 2017) and append the first paragraph of the article to the input.

5 Experiments

5.1 Experimental Settings

Datasets We evaluate the calibration performance on both multiple-choice QA datasets and extractive QA datasets listed in Table 2. To test whether our calibration methods can generalize to out-of-domain datasets, we use a subset of datasets of multiple-choice/extractive QA to train our methods, and the remaining subset of datasets to evaluate the performance. Specifically, we use ARC (easy), AI2 Science Question (elementary), OpenbookQA, QASC, Winogrande, CommonsenseQA, and PhysicalIQA as the training subset for multiple-choice QA (denoted as **MC-train**), and SQuAD 1.1, NewsQA as the training subset for extractive QA (denoted as **Ext-train**). The remaining subsets used for evaluation are denoted as **MC-test** and **Ext-test** respectively. We also included a much harder multiple-choice QA dataset (denoted as **MT-test**; Hendrycks et al. (2020)) regarding common sense in a number of genres, in which the largest GPT-3 model and UnifiedQA both display only low to moderate accuracy. For fine-tuning methods, we use the train split of MC-train/Ext-train to fine-tune the LMs. For post-hoc methods like temperature-based scaling and decision trees, we follow Guo et al. (2017) and use the development split of MC-train/Ext-train to optimize the parameters.[3]

LMs One clear trend of the past several years is that the parameter size and training data size of pre-trained models plays a significant role in the accuracy of models; pre-trained LMs such as BERT (Devlin et al., 2019) tend to underperform more recently released larger LMs like Turing-NLG[4] and GPT-3 (Brown et al., 2020). Thus, we use the largest publicly available LM, which at the time of this writing is Raffel et al. (2019)'s T5 model. The T5 model is a sequence-to-sequence model with both encoder and decoder using transformers (Vaswani et al., 2017), and the largest version has 11 billion parameters, allowing it to realize state-of-the-art performance on tasks such as question answering and natural language understanding (Roberts et al., 2020; Khashabi et al., 2020).

Specifically, we use two varieties of this model. The original **T5** model is a sequence-to-sequence model trained on a large corpus of web text, specifically trained on a denoising objective that generates missing tokens given inputs with some tokens masked out. The **UnifiedQA** model, uses the initial T5 model and fine-tunes on a variety of QA datasets by converting multiple-choice, extractive QA formats into a unified sequence-to-sequence format, similar to the one that we show in Table 1. We use the 3-billion versions in our main experiments in subsection 5.3 (for efficiency purposes), but also report the performance of the largest 11-billion versions in ablation studies subsection 5.5.

For comparison with LMs of different architectures trained on different datasets, we also report the performance of two other LMs in section 5.5: the 0.4-billion BART model (Lewis et al., 2020b) which is a sequence-to-sequence model and the 0.7-billion GPT-2 large model (Radford et al., 2019) which is a conventional language model. We fine-tune them following the same recipe of UnifiedQA (Khashabi et al., 2020).

Evaluation Metrics We use accuracy to measure the prediction performance of our methods, and ECE to measure the calibration performance. Accuracy is computed as the ratio of question-answer pairs for which the correct answer has the highest probability among all the candidates in $\mathcal{I}(x)$. ECE is computed using Equation 2 by bucketing all candidate answers in $\mathcal{I}(x)$ based on confidence. For MC-test and Ext-test which include multiple datasets, we compute accuracy and ECE on each dataset separately and average across them to avoid the metrics being dominated by large datasets.

Implementation Details We fine-tune UnifiedQA-3B with a batch size of 16 for 3k steps and UnifiedQA-11B with a batch size of

[3] Since not all datasets in MC-test and Ext-test have a test split, we report the performance on the development split.

[4] https://msturing.org/

3 for 15k steps on a v3-8 TPU. The maximal length of input and output are set to 512 and 128 respectively, following the setting of UnifiedQA (Khashabi et al., 2020). For extractive QA datasets, we use top $R = 10$ first tokens and finally $K = 5$ spans are used as candidates. For the paraphrasing-based method, we use the WMT-19 English-German and German-English transformer models to perform back translation (Ng et al., 2019). The beam size is set to 10 for both directions, which will yield $10 \times 10 = 100$ paraphrases in the end. Since some paraphrases are duplicated, we count the frequency and use the top 5 unique paraphrases in our main experiments subsection 5.3. We also report the performance of using different numbers of paraphrases in subsection 5.5. For the retrieval-based augmentation, we use the KILT toolkit (Petroni et al., 2020b) to retrieve the most relevant article from the Wikipeida dump, and append the first three sentences of the first paragraph of the retrieved article to the input. For the feature-based decision trees model, we use XGBoost (Chen and Guestrin, 2016) with logistic binary objective, max depth of 4, number of parallel trees of 5, and subsample ratio of 0.8. We use **Temp.** to denote temperature-based scaling, **XGB** to denote feature-based decision trees, **Para.** to denote paraphrasing, **Aug.** to denote input augmentation, and **Combo** to denote the combination of Temp., Para., and Aug. in the experimental section. We use the model with the best calibration performance in post-hoc calibration experiments. For multiple-choice QA, we use the UnifiedQA model after margin-based fine-tuning. For extractive QA, we use the original UnifiedQA model.

5.2 Are LM-based QA Models Well Calibrated?

As shown in Table 4, our baseline models (i.e., T5 and UnifiedQA) are strong, achieving state-of-the-art accuracy on a diverse range of QA datasets. On the MT-test datasets, the UnifiedQA model even outperforms the largest version of GPT-3 with 175 billions parameters (Hendrycks et al., 2020). Despite the impressive performance, these models are not well calibrated, with ECE higher than 0.2 on the MT-test dataset. We found that LMs tend to be over-confident about cases they do not know, as shown in the confidence distribution in the first row of Figure 2 that most predictions have aggressive confidence being close to 0 or 1. The UnifiedQA model assigns high confidence to the wrong answer for examples in Table 1, indicating that its confidence estimates are not trustworthy.

5.3 Can LM-based QA Models be Calibrated?

We calibrate the UnifiedQA model using both fine-tuning-based methods and post-hoc methods and show their performance in Table 4 and Table 5 respectively.

Overall, on multi-choice QA datasets (i.e., MC-test and MT-test), both fine-tuning-based methods and post-hoc methods can improve ECE while maintaining accuracy compared to the baseline UnifiedQA model. The best-performing method (i.e., Combo), which combines margin-based fine-tuning, temperature-based scaling, paraphrasing, and input augmentation, improves ECE from 0.095 to 0.044 by over 53%. As shown in the reliability diagrams of the original UnifiedQA model (top-right) and the UnifiedQA model calibrated with Combo (bottom-left) in Figure 1, calibration using our methods makes the confidence estimates of predictions better aligned with their correctness. Comparing those two diagrams, an interesting observation is that our method seems to over-calibrate the LM, making over-estimated bars on the right-hand side of the top-right diagram (bars lower than the diagonal) under-estimated and vice versa. This is probably caused by the temperature being too aggressive (i.e., too large), making the distribution too flat. Note that the datasets used to learn the temperature (MC-train) and used in evaluation (MC-test) are different, which we hypothesize is the reason why the temperature is too aggressive. We verify this by learning an oracle temperature on the evaluation datasets (MC-test). The learned temperature indeed becomes smaller ($1.35 \rightarrow 1.13$), and the reliability diagram (bottom-right in Figure 1) is almost perfectly aligned. This demonstrates the challenge of calibrating LMs across different domains.

However, on extractive QA datasets, the improvement brought by different calibration methods is smaller. We hypothesize that this is because the candidate set $\mathcal{I}(X)$ generated by the span-based decoding method for extractive QA are harder to calibrate than the manually curated candidate answers for multiple-choice QA. We compute the average entropy of the confidence of the

Method	MC-test		MT-test		Ext-test	
	ACC	ECE	ACC	ECE	ACC	ECE
T5	0.313	0.231	0.268	0.248	0.191	0.166
UnifiedQA	0.769	0.095	0.437	0.222	0.401	0.114
+ softmax	0.767	0.065	0.433	0.161	0.394	**0.110**
+ margin	0.769	**0.057**	0.431	**0.144**	0.391	0.112

Table 4: Performance of different fine-tuning methods.

Method	MC-test		MT-test		Ext-test	
	ACC	ECE	ACC	ECE	ACC	ECE
Baseline	0.769	0.057	0.431	0.144	0.401	0.114
+ Temp.	0.769	0.049	0.431	**0.075**	0.401	0.107
+ XGB	0.771	0.055	0.431	0.088	0.402	**0.103**
+ Para.	0.767	0.051	0.429	0.122	0.393	0.114
+ Aug.	0.744	0.051	0.432	0.130	0.408	0.110
+ Combo	0.748	**0.044**	0.431	0.079	0.398	0.104

Table 5: Performance of different post-hoc methods using the UnifiedQA model after margin-based fine-tuning or the original UnifiedQA model as the baseline model. "+Combo" denotes the method using both Temp., Para., and Aug.

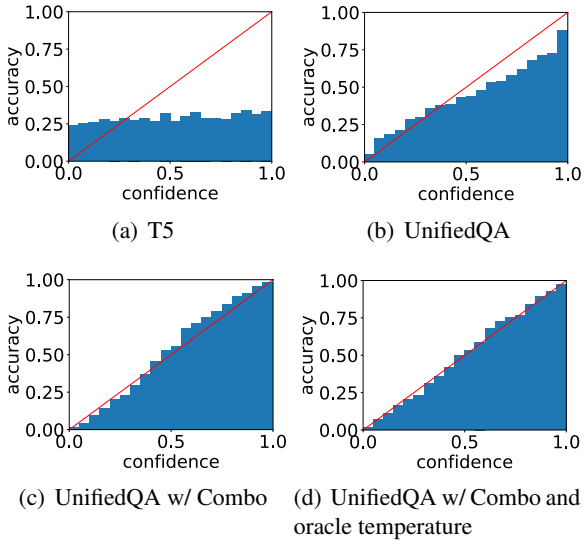
UnifiedQA model over $\mathcal{I}(X)$ on both extractive QA (Ext-test) and multiple-choice QA datasets (MC-test), and found that Ext-test indeed has much higher entropy compared to MC-test (0.40 vs 0.13), which partially explains the difficulty of calibration on extractive QA datasets.

5.4 Analysis of Individual Calibration Methods

In this section, we discuss each method in detail and analyze why they can improve calibration performance.

Objective Function Matters. The original UnifiedQA model is fine-tuned based on MLE, which maximizes the probability of the gold answer given the question. Both softmax-based and margin-based fine-tuning, which explicitly compare and adjust the probability of candidate answers, can further improve ECE on multiple-choice datasets. We argue that the softmax-based and margin-based objective functions are better suited for questions with potential candidates.

Post-processing Confidence is Effective Universally. Post-processing the raw confidence either solely based on confidence or other features is effective across all datasets, which is consistent with the conclusion on other tasks such as structured



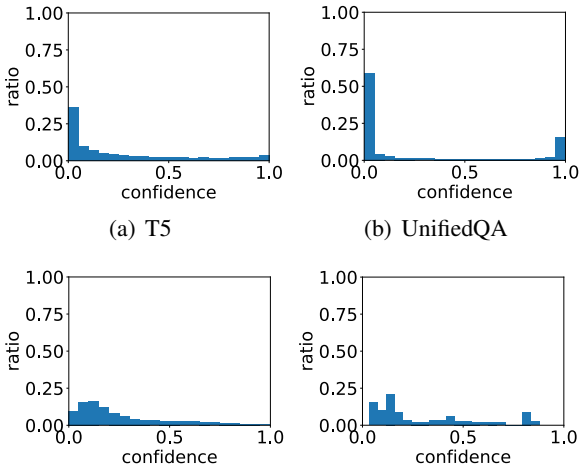
(a) T5 (b) UnifiedQA

(c) UnifiedQA w/ Combo (d) UnifiedQA w/ Combo and oracle temperature

Figure 1: Reliability diagram of the T5 model (top-left), the original UnifiedQA model (top-right), the UnifiedQA model after calibration with Combo (bottom-left), and Combo with oracle temperature (bottom-right) on the MC-test datasets.

prediction and natural language inference (Jagannatha and Yu, 2020; Desai and Durrett, 2020). We demonstrate the histogram of confidence before and after applying temperature-based scaling or feature-based decision trees in Figure 2. LMs tend to be over-confident, with most predictions having either extremely high or low confidence. Both methods can successfully re-scale the confidence to reasonable ranges, thus improving the calibration performance.

Paraphrasing Answers and Input Augmentation can Improve Confidence Estimation. The improvement brought by using paraphrasing is significant on multiple-choice datasets, demonstrating that using diverse expressions can indeed improve confidence estimation. To better understand under what circumstances paraphrasing works, we group candidate answers into two categories: the first group includes candidate answers that become better calibrated using paraphrases; the second group includes candidate answers whose confidence remains the same using paraphrases. We say that a candidate becomes better calibrated if its confidence increases/decreases by 20% if it is a correct or incorrect answer respectively. We found that the average length of questions for better calibrated candidates (187) is much shorter than that of candidates without improvement (320), indicating that paraphrasing is



(a) T5 (b) UnifiedQA

(c) UnifiedQA w/ Temp. (d) UnifiedQA w/ XGB

Figure 2: The ratio of predictions with respect to confidence of the T5 model (top-left), the UnifiedQA model (top-right), the UnifiedQA model after temperature-based calibration (bottom-left), and the UnifiedQA model after feature-based calibration (bottom-right) on the MC-test datasets.

useful mainly for short questions. We also compute the diversity of word usage in paraphrases using the number of unique words divided by the total length of paraphrases. We found that better calibrated candidates have slightly higher diversity (0.35 vs 0.32), which is consistent with our intuition. Retrieval-based augmentation can also improve calibration performance on multiple-choice datasets, which is probably because the retrieved documents can provide extra evidence about the question, making LMs more robust at confidence estimation.

Calibration Methods are Complementary. By combining margin-based fine-tuning, temperature-based scaling, paraphrasing, and input augmentation, we achieve the best ECE on MC-test, demonstrating that these calibration methods are complementary to each other.

5.5 Ablation Study

In this section, we perform an ablation study to examine different aspects of LM calibration, including calibration performance of different LMs, across LMs with different sizes, using different numbers of paraphrases, and across datasets with potential domain shift.

Performance of Different LMs. We report the performance of two other LMs in Table 6. Both the BART and GPT-2 models are smaller than T5, thus the overall accuracy and calibration performance are lower than that of T5. Both fine-tuning and post-hoc calibration methods can improve ECE, indicating that our methods are applicable to LMs trained with different datasets and architectures.

Method	BART		GPT-2 large	
	ACC	ECE	ACC	ECE
Original	0.295	0.225	0.272	0.244
+ UnifiedQA	0.662	0.166	0.414	0.243
+ softmax	0.658	0.097	0.434	0.177
+ margin	0.632	0.090	0.450	0.123
+ Temp.	0.632	**0.064**	0.450	**0.067**
+ XGB	0.624	0.090	0.440	0.080
+ Para.	0.624	0.084	0.436	0.104
+ Aug.	0.600	0.089	0.441	0.126
+ Combo	0.591	0.065	0.429	0.069

Table 6: Performance of different LMs on the MC-test dataset. "Original" indicates the original language model, and "+ UnifiedQA" indicates fine-tuning following the recipe of UnifiedQA.

Performance of LMs with Different Sizes. We conduct experiments using the largest version (i.e., 11B) of the T5 and UnifiedQA model to analyze how calibration performance varies with respect to the size of the LM in Table 7. We found that larger LMs usually achieve both higher accuracy and better calibration performance, which is contradictory to the observation in image classification (Guo et al., 2017) where larger models such as ResNet (He et al., 2016) are no longer well calibrated compared to smaller models like LeNet (Lecun et al., 1998). Given the fact the size of both the pre-training corpus and LMs are extremely larger compared to previous practice, we might have completely different observations with respect to confidence estimation. Unlike ResNet trained on CIFAR-100, the training of LMs is not bottlenecked by the dataset, and larger LMs have a stronger capacity to model text distribution and memorize facts, which leads to better calibration performance overall (Kaplan et al., 2020). Overall, our methods can improve ECE from 0.067 to 0.032 using the 11B UnifiedQA model on the MC-test dataset, and from 0.175 to 0.085 on the MT-test dataset. However, compared to the 3B version, improvement brought by post-hoc calibration methods is smaller, which is probably because the 11B version is better optimized and more

Method	MC-test ACC	MC-test ECE	MT-test ACC	MT-test ECE
T5	0.359	0.206	0.274	0.235
UnifiedQA	0.816	0.067	0.479	0.175
+ softmax	0.823	0.041	0.488	0.129
+ margin	0.819	0.034	0.485	0.107
+ Temp.	0.819	0.036	0.485	0.098
+ XGB	0.818	0.065	0.486	0.108
+ Para.	0.820	0.035	0.484	0.092
+ Aug.	0.812	**0.031**	0.493	0.090
+ Combo	0.807	0.032	0.494	**0.085**

Table 7: Performance of the 11B LMs.

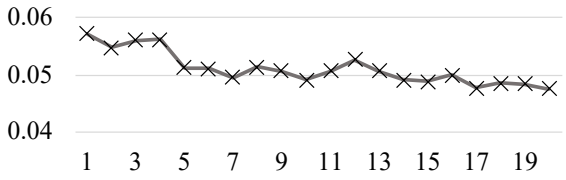


Figure 3: ECE of the UnifiedQA model using different numbers of paraphrases on the MC-test datasets.

knowledgeable.

Performance using Different Numbers of Paraphrases. In Figure 3, we experiment with different numbers of paraphrases using the UnifiedQA model on MC-test datasets. The overall trend is that the more paraphrases we use, the better calibrated the LM, demonstrating that using different variations to express the candidate answer can improve confidence estimation. The improvements using more than 10 paraphrases are subtle, so 5-10 paraphrases may represent a good trade-off between computational cost and performance in practical settings.

Performance on Training and Evaluation Datasets. As introduced in the experimental section, we perform calibration on the MC-train dataset and evaluate the final performance on the MC-test dataset to study whether our calibration methods can generalize to out-of-domain dataset. We compare the performance on the training dataset and the evaluation dataset in Table 8. We found that on both datasets, each individual method can improve ECE, indicating that our method can generalize to out-of-domain datasets. Note that the improvement on the training dataset ($0.133 \rightarrow 0.042$) is larger than on improvement on the evaluation dataset ($0.095 \rightarrow 0.044$), which is probably caused by the domain shift between the two datasets.

Method	MC-train ACC	MC-train ECE	MC-test ACC	MC-test ECE
T5	0.334	0.228	0.313	0.231
UnifiedQA	0.727	0.133	0.769	0.095
+ softmax	0.735	0.084	0.767	0.065
+ margin	0.737	0.069	0.769	0.057
+ Temp.	0.737	0.051	0.769	0.049
+ XGB	0.737	0.074	0.771	0.055
+ Para.	0.742	0.053	0.767	0.051
+ Aug.	0.721	0.059	0.744	0.051
+ Combo	0.722	**0.042**	0.748	**0.044**

Table 8: Performance comparison between training and evaluation datasets.

6 Related Work

Calibration Calibration is a well-studied topic in other tasks such as medical diagnosis (Jiang et al., 2012) and image recognition (Guo et al., 2017; Lee et al., 2018). Previous works in NLP have examined calibration in structured prediction problems such as part-of-speech tagging and named entity recognition (Jagannatha and Yu, 2020), natural language understanding tasks such as natural language inference, paraphrase detection, extractive question answering, and text classification (Desai and Durrett, 2020; Kamath et al., 2020; Kong et al., 2020). In contrast, we focus on calibrating LMs themselves by treating them as natural language generators that predict the next words given a particular input.

LM probing Previous works probe pre-trained LMs with respect to syntactic and semantic properties (Hewitt and Manning, 2019; Tenney et al., 2019), factual knowledge (Petroni et al., 2019; Poerner et al., 2019; Jiang et al., 2020b), commonsense knowledge (Trinh and Le, 2018; Kocijan et al., 2019), and other properties (Talmor et al., 2019a). These works usually focus on what LMs know, while in this paper we also consider the cases when LMs do not know the answer with confidence.

7 Conclusion

In this paper, we examine the problem of calibration in LMs used for QA tasks. We first note that despite the impressive performance state-of-the-

art LM-based QA models tend to be poorly calibrated in their probability estimates. To alleviate this problem, we attempted several methods to either fine-tune the LMs, or adjust the confidence by post-processing raw probabilities, augmenting inputs, or paraphrasing candidate answers. Experimental results demonstrate the effectiveness of these methods. Further analysis reveals the challenges of this problem, shedding light on future work on calibrating LMs.

Some future directions could be developing calibration methods for LMs on a more fine-grained level than simply holistic calibration across the entire dataset. For example, there has been significant interest in how models perform across diverse subsets of the entire training data (Hashimoto et al., 2018) and how they reflect dataset biases (Rudinger et al., 2018), and the interaction of model confidence with these phenomena is of significant interest. It is also interesting to investigate the effect of calibration on users or downstream tasks. For instance, providing users with model confidences can influence downstream decisions (Zhang et al., 2020), and users may want to adjust required confidence thresholds on critical domains (e.g., health, safety, medicine). All of these are interesting paths of inquiry for future research.

Acknowledgements

This work was supported in part by a gift from Bosch research. The authors thank the Google Cloud and TensorFlow Research Cloud for computation credits that aided in the execution of this research.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Çelikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics.

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, New York, USA.

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. *CoRR*, abs/2005.14165.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM.

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Second Conference on Applied Natural Language Processing*, pages 136–143, Austin,

Texas, USA. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5924–5931. Association for Computational Linguistics.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. *CoRR*, abs/2003.07892.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 743–753. Association for Computational Linguistics.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 946–958. Association for Computational Linguistics.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Tatsunori B. Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness without demographics in repeated loss minimization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1934–1943. PMLR.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *CoRR*, abs/2009.03300.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Abhyuday Jagannatha and Hong Yu. 2020. Calibrating structured output predictors for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2078–2092. Association for Computational Linguistics.

Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado. 2012. Calibrating predictive model estimates to support personalized medicine. *J. Am. Medical Informatics Assoc.*, 19(2):263–274.

Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. X-factr: Multilingual factual knowledge retrievalfrom pretrained language models.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020b. How can we know what language models know? *Transactions of the Association for Computational Linguistics (TACL)*.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5684–5696. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single QA system. *CoRR*, abs/2005.00700.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8082–8090. AAAI Press.

Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. 2019. A surprisingly robust trick for the winograd schema challenge. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4837–4842. Association for Computational Linguistics.

Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. 2020. Calibrated language model fine-tuning for in- and out-of-distribution data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1326–1340. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 785–794. Association for Computational Linguistics.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020a. Pre-training via paraphrasing. *CoRR*, abs/2006.15020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman

Goyal, Heinrich Küttler, Mike Lewis, Wentau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020c. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 58–62. Association for Computational Linguistics.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 314–319. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. *arXiv preprint arXiv:1803.00047*.

Fabio Petroni, Patrick S. H. Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020a. How context affects language models' factual predictions. *CoRR*, abs/2005.04611.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2020b. Kilt: a benchmark for knowledge intensive language tasks. In *arXiv:2009.02252*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. E-bert: Efficient-yet-effective entity embeddings for bert. *CoRR*, abs/1911.03681.

Katyanna Quach. 2020. Researchers made an OpenAI GPT-3 medical chatbot as an experiment. it told a mock patient to kill themselves. *The Register*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+

questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Matthew Richardson, Christopher J. C. Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 193–203. ACL.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *CoRR*, abs/2002.08910.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 8–14. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4462–4472. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2019a. olmpics - on what language model pre-training captures. *CoRR*, abs/1912.13283.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019b. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4149–4158. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. *CoRR*, abs/1806.02847.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 5306–5314. Association for Computational Linguistics.

Yunfeng Zhang, Q. Vera Liao, and Rachel K. E. Bellamy. 2020. Effect of confidence and explanation on accuracy and trust calibration in ai-assisted decision making. In *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 295–305. ACM.

Pei Zhou, Rahul Khanna, Bill Yuchen Lin, Daniel Ho, Xiang Ren, and Jay Pujara. 2020. Can BERT reason? logically equivalent probes for evaluating the inference capabilities of language models. *CoRR*, abs/2005.00782.

How Many Data Points is a Prompt Worth?

Teven Le Scao
Hugging Face
teven@huggingface.co

Alexander M. Rush
Hugging Face
sasha@huggingface.co

Abstract

When fine-tuning pretrained models for classification, researchers either use a generic model *head* or a task-specific *prompt* for prediction. Proponents of prompting have argued that prompts provide a method for injecting task-specific guidance, which is beneficial in low-data regimes. We aim to quantify this benefit through rigorous testing of prompts in a fair setting: comparing prompted and head-based fine-tuning in equal conditions across many tasks and data sizes. By controlling for many sources of advantage, we find that prompting does indeed provide a benefit, and that this benefit can be quantified per task. Results show that prompting is often worth 100s of data points on average across classification tasks.

1 Introduction

The main paradigm for adapting pretrained models for classification (Radford, 2018; Dong et al., 2019; Devlin et al., 2018) is fine-tuning via an explicit classifier head. However, an alternative approach has arisen: adapting the pretrained language model directly as a predictor through autoregressive text generation (Radford et al., 2019) or completion of a cloze task (Trinh and Le, 2018). This method is notably used in T5 fine-tuning (Raffel et al., 2019) leading to state-of-the-art results on the SuperGLUE benchmark (Wang et al., 2019).

One argument made for classification by direct language generation is that it allows us to pick custom *prompts* for each task (McCann et al., 2018). While this approach can be used for zero-shot classification (Puri and Catanzaro, 2019) or priming (Brown et al., 2020), it can also be used in fine-tuning to provide extra task information to the classifier, especially in the low-data regime (Schick and Schütze, 2020a,b).

Code available at https://github.com/TevenLeScao/pet

If this argument is indeed true, it is natural to ask how it impacts the sample efficiency of the model, or more directly, *how many data points is a prompt worth?* As with many low-data and pretraining-based problems, this question is complicated by the fine-tuning setup, training procedure, and prompts themselves. We attempt to isolate these variables through diverse prompts, multiple runs, and best practices in low-training data fine-tuning. We introduce a metric, the *average data advantage*, for quantifying the impact of a prompt in practice.

Our experiments find that the impact of task-targeted prompting can nicely be quantified in terms of direct training data, and that it varies over the nature of different tasks. On MNLI (Williams et al., 2018), we find that using a prompt contributes approximately 3500 data points. On SuperGLUE, it adds approximately 280 data points on RTE (Dagan et al., 2005) and up to 750 on BoolQ (Clark et al., 2019). In low- to medium-data settings, this advantage can be a real contribution to training a model.

2 Related Work

Prompting has been used both for zero-shot and fine-tuning based methods. Zero-shot approaches attempt to answer a task with a prompt without fine-tuning through generation (Radford et al., 2019). GPT3 (Brown et al., 2020) extends this approach to a supervised priming method by taking in training data as priming at inference time, so it can attend to them while answering. T5 (Raffel et al., 2019) and other sequence-to-sequence pretrained models use standard word-based fine-tuning with a marker prompt to answer classification tasks with strong empirical success. Our setting differs in that we are interested in using task-based prompts and fine-tuning, in-between the T5 and GPT2 setting.

Our setting most closely resembles PET (Schick and Schütze, 2020a,b), which claims that task-specific prompting helps transfer learning, espe-

cially in the low-data regime. However, in order to reach the best possible results on SuperGLUE, PET introduces several other extensions: semi-supervision via additional pseudo-labeled data, ensembling models trained with several different prompts, and finally distilling the ensemble into a linear classifier rather than a language model. Our aim is to isolate the specific contributions of prompting within supervised fine-tuning.

Finally, recent papers have experimented with discovering prompts through automated processes tailored to the language model (Jiang et al., 2020; Schick et al., 2020). We limit ourselves to human-written prompts, as we are interested into whether prompting itself specifically adds information to the supervised task. It is an interesting question as to whether automatic prompts can have this same impact (relative to the training data they require).

3 Comparison: Heads vs Prompts

Consider two transfer learning settings for text classification: *head-based*, where a generic head layer takes in pretrained representations to predict an output class; *prompt-based*, where a task-specific pattern string is designed to coax the model into producing a textual output corresponding to a given class. Both can be utilized for fine-tuning with supervised training data, but prompts further allow the user to customize patterns to help the model.

For the *prompt* model we follow the notation from PET (Schick and Schütze, 2020a) and decompose a prompt into a *pattern* and a *verbalizer*. The *pattern* turns the input text into a cloze task, i.e. a sequence with a masked token or tokens that need to be filled. Let us use as example an excerpt from SuperGLUE task BoolQ (Clark et al., 2019), in which the model must answer yes-or-no binary questions. In order to let a language model answer the question in *italics*, our pattern is in **bold** (Schick and Schütze, 2020b):

> "Posthumous marriage – Posthumous marriage (or necrogamy) is a marriage in which one of the participating members is deceased. It is legal in France and similar forms are practiced in Sudan and China. Since World War I, France has had hundreds of requests each year, of which many have been accepted. **Based on the previous passage, *can u marry a dead person in france ?* <MASK>**"

The masked word prediction is mapped to a *verbalizer* which produces a class. (here "Yes": True. "No": False[1]). Several *pattern-verbalizer pairs* (*PVPs*) could be used for a single task, differing either through the pattern, the verbalizer, or both. Fine-tuning is done by training the model to produce the correct verbalization. The loss is the cross-entropy loss between the correct answer and the distribution of probabilities amongst the tokens in the verbalizer. We re-use pattern choices from Schick and Schütze (2020b); examples are available in Appendix A.

4 Experimental Setting

We run all experiments with the same pretrained checkpoint, *roberta-large* (355M parameters) from RoBERTa (Liu et al., 2019), which we load from the *transformers* (Wolf et al., 2020) library.[2] In line with previous observations (McCoy et al., 2019; Dodge et al., 2020; Lee et al., 2020), head-based fine-tuning performance varies considerably. We follow recommendations of Mosbach et al. (2020) and Zhang et al. (2020) to train at a low learning rate (10^{-5}) for a large number of steps (always at least 250, possibly for over 100 epochs).

We perform our evaluation on SuperGLUE and MNLI (Williams et al., 2018). These datasets comprise a variety of tasks, all in English, including entailment (MNLI, RTE (Dagan et al., 2005), CB (de Marneffe et al., 2019)), multiple choice question answering (BoolQ (Clark et al., 2019), MultiRC (Khashabi et al., 2018)), and commonsense reasoning (WSC (Levesque et al., 2012), COPA (Roemmele et al., 2011), WiC (Pilehvar and Camacho-Collados, 2018)). We do not include ReCoRD (Zhang et al., 2018) in our comparisons as there is no head model to compare with, since it is already a cloze task. Data sizes range from 250 data points for CB to 392,702 for MNLI. As test data is not publicly available for SuperGLUE tasks, we set aside part of training (from 50 for CB, COPA and MultiRC to 500 for BoolQ) to use for development, and evaluate on their original validation sets. For MNLI, we use the available matched validation and test sets.

We compare models across a scale of available data, starting with 10 data points and increasing exponentially (as high-data performance tends to

[1] The correct answer here is, of course, *yes*. Originated in 1803 as Napoleon rose to power, this practice was mainly to the benefit of war widows.

[2] After experimenting with RoBERTa, AlBERT (Lan et al., 2019) and BERT (Devlin et al., 2018), we found *roberta-large* to have the most consistent performance.

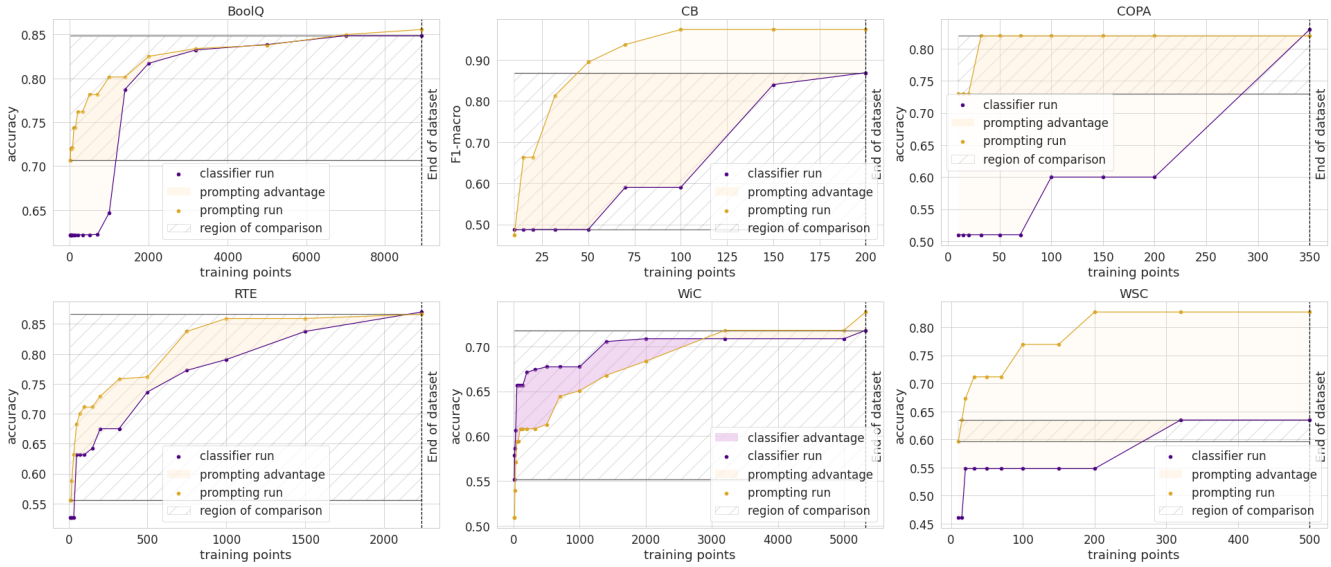


Figure 1: Prompting vs head (classifier) performance across data scales, up to the full dataset, for six SuperGLUE tasks. Compares the best prompt and head performance at each level of training data across 4 runs. Highlighted region shows the accuracy difference of the models. Cross-hatch region highlights the lowest- and highest- accuracy matched region in the curves. The highlighted area in this region is used to estimate the data advantage.

saturate) to the full dataset. For example, for MultiRC, which has 969 data points initially, we start by reserving 50 data points for development. This leaves us with 919 training points, and we train models with 10, 15, 20, 32, 50, 70, 100, 150, 200, 320, 500, 750, and 919 training points. We run every experiment 4 times in order to reduce variance, for a total of 1892 training runs across all tasks. At every point, we report the best performance that has been achieved at that amount of data or lower. Full graphs are presented in Appendix B.

5 Results

Figure 1 shows the main results comparing head- and prompt-based fine-tuning with the best-performing pattern on that task. Prompting enjoys a substantial advantage on every task, except for WiC as is reported in previous results (Schick and Schütze, 2020b). Both approaches improve with more training data, but prompting remains better by a varying amount. Many tasks in SuperGLUE have relatively few data points, but we also see an advantage in large datasets like BoolQ and MNLI.

To quantify how many data points the prompt is worth, we first isolate the y-axis band of the lowest- and highest- accuracy where the two curves match in accuracy.[3] The horizontal line at these points represents the advantage of prompting. We then take the integral in this region, i.e. area between the linearly-interpolated curves[4], divided by the height of the band. The area has the dimension of a quantity of data points times the metric unit, so dividing by the performance range yields a # of data points advantage. As low data training is sensitive to noise, in addition to following best training practices we run several different experiments for each x-point. We use a bootstrapping approach to estimate confidence over these runs. Specifically, we hold out one of the 4 head runs and 4 prompt runs (16 combinations total), and compute the standard deviation of those outcomes.

We report these quantities for every task in Table 1 as *Average advantage*. For almost all the tasks, we see that prompting gives a substantial advantage in terms of data efficiency, adding the equivalent of hundreds of data points on average.

6 Analysis

Impact of Pattern vs Verbalizer The intuition of prompts is that they introduce a task description in natural language, even with few training points. To better understand the zero-shot versus adaptive nature of prompts, we consider a *null verbalizer*, a control with a verbalizer that cannot yield semantic information without training. For every task that requires filling in one word (which excludes

[3] We assume asymptotically the two curves would match, but are limited by data.

[4] In areas where the head model is better, if any, get subtracted from the total.

	Average Advantage (# Training Points)							
	MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
P vs H	3506 ± 536	752 ± 46	90 ± 2	288 ± 242	384 ± 378	282 ± 34	-424 ± 74	281 ± 137
P vs N	150 ± 252	299 ± 81	78 ± 2	-	74 ± 56	404 ± 68	-354 ± 166	-
N vs H	3355 ± 612	453 ± 90	12 ± 1	-	309 ± 320	-122 ± 62	-70 ± 160	-

Table 1: Average prompting advantage in number of data points for MNLI & SuperGLUE tasks. *P* denotes the prompt model, *H* the head model. On average across performance levels, an MNLI prompt model yields the results of an MNLI head model trained with 3500 additional data points. Confidence levels are based on a multiple random runs (see text). *N* indicates a null-verbalizer prompting task that replaces the verbalizer with a non-sensical mapping. *The comparison band of MultiRC is too small as the head baseline fails to learn beyond majority class; we use the full region for a lower-bound result.

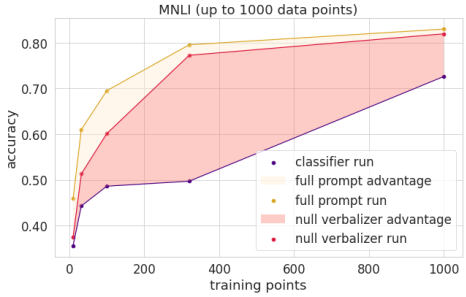


Figure 2: Comparison of full prompt and null verbalizer advantage on MNLI at lower data scales.

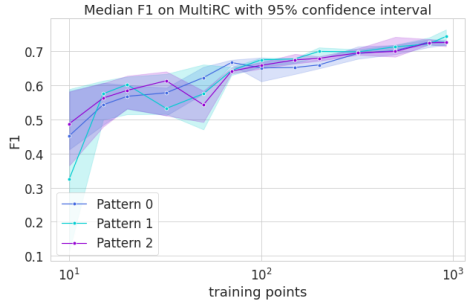


Figure 3: Median performance on MultiRC across runs for three prompts. Differences are inconsistent and eclipsed by the variance within one prompt's runs.

the more free-form COPA and WSC), we replace the verbalizers, for example, "yes", "no", "maybe", "right" or "wrong", with random first names.

Table 1 shows the advantage of the standard prompt over the null verbalizer to estimate this control. We see that for small data tasks such as CB, the null verbalizer removes much of the benefits of prompting. However, with more training data, the model seems to adapt the verbalizer while still gaining the inductive bias benefits of the pattern. Figure 2 showcases this dynamic on MNLI. This result further shows that prompting yields data efficiency even if it is not directly analogous to the generation process of training.

Impact of Different Prompts If the prompt acts as a description of the task, one would expect different valid descriptions to vary in their benefits. In order to compare the different prompts we used on each task, we chart the median performance for each of them under different runs. In nearly every experiment, we find that the confidence intervals of those curves largely overlap, implying that prompt choice is not a dominant hyperparameter, i.e. the variance across random seeds usually outweighs the possible benefits of prompt choice. One exception is the low-data regime of BoolQ, where one of the prompts enjoys a significant few-shot advantage over the others. We plot this curve for MultiRC in Figure 3 and the rest in Appendix C.

Metric sensitivity We treat each metric linearly in calculating advantage; alternatively, we could reparameterize the y axis for each task. This choice does not have a consistent effect for or against prompting. For example, emphasizing gains close to convergence increases prompting advantage on CB and MNLI but decreases it on COPA or BoolQ.

7 Conclusion

We investigate prompting through a systematic study of its data advantage. Across tasks, prompting consistently yields a varying improvement throughout the training process. Analysis shows that prompting is mostly robust to pattern choice, and can even learn without an informative verbalizer. On large datasets, prompting is similarly helpful in terms of data points, although they are less beneficial in performance. In future work, we hope to study the mechanism and training dynamics of the prompting benefits.

8 Impact statement

Significant compute resources were used to run this paper's experiments. A single experiment (defined as one model run, at one data level, on one task) was quite light-weight, taking usually a little under an hour on a single Nvidia V100. However, as we computed a little under two thousand runs, this adds up to about 1800 GPU hours, to which one must add around 400 GPU hours of prototyping and hyper-parameter searching. Those 2200 GPU hours would usually have necessitated the release of about 400kg of CO2, about 40% of a transatlantic flight for a single passenger, in the country where we ran the experiments, although we used a carbon-neutral cloud compute provider.

The main benefit of prompting, rather than compute efficiency, is data efficiency. Although we ran all of our experiments on English, we hope that this property will be especially helpful in low-resource language applications. In a sense, a practitioner could then remedy the lack of task-specific data in their language by introducing information through a prompt. However, this comes with the inherent risk of introducing human biases into the model. Prompt completion also suffers from biases already present within the language model (Sheng et al., 2019). This could cause a prompted model to repeat those biases in classification, especially in the few-shot setting where prompting mostly relies on the pretrained model.

9 Acknowledgments

We thank Steven Cao and Joe Davison for the discussions about prompting that initially spurred this paper. We further thank Timo Schick for making the code for PET available and for discussions about performance replication. We lastly thank Canwen Xu, Yacine Jernite, Victor Sanh, Dimitri Lozeve and Antoine Ogier for their help with the figures and writing of this draft.

References

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.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *CoRR*, abs/1905.10044.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. pages 177–190.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know?

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 252–262. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*. AAAI Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2019. Berts of a feather do not generalize together: Large variability in generalization across models with similar test set performance.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines.

Mohammad Taher Pilehvar and José Camacho-Collados. 2018. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *CoRR*, abs/1808.09121.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *CoRR*, abs/1912.10165.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few shot text classification and natural language inference.

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

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. *CoRR*, abs/1909.01326.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning.

A Choice of prompts

We use a subset of prompts from (Schick and Schütze, 2020b).

- For entailment tasks (**CB**, **MNLI**, **RTE**) given a premise p and hypothesis h:

  h? | <MASK>, p

  "h"? | <MASK>. "p"

  with "yes" as a verbalizer for entailment, "no" for contradiction, "maybe" for neutrality.

- For **BoolQ**, given a passage p and question q:

  p. Question: q? Answer: <MASK>.

  p. Based on the previous passage, q? <MASK>.

  Based on the following passage, q? <MASK>. p

  with "yes" or "no" as verbalizers for True and False.

- For **COPA**, given an effect e and possible causes c_1 and c_2:

  "c_1" or "c_2"? e, so <MASK> .

  c_1 or c_2? e, so <MASK>.

  and a cause c and possible effects e_1 and e_2:

  "e_1" or "e_2"? <MASK>, because c.

  e_1 or e_2? <MASK>, because c.

  and the verbalizer is the identity function.

- For **MultiRC**, given a passage p, question q and answer a, we estimate whether a is a proper answer with:

  p. Question: q? Is it a? <MASK>.

  p. Based on the previous passage, q? Is the correct answer a? <MASK>.

  p. Question: q? Is the correct answer a? <MASK>.

- For **WiC**, given two sentences s_1 and s_2 and a word w, we classify whether w was used in the same sense.

  "s_1" / "s_2". Similar sense of "w"? <MASK>.

  s_1 s_2 Does w have the same meaning in both sentences? <MASK>.

  With "yes" and "no" as verbalizers.

- For **WSC**, given a sentence s with a marked pronoun *p* and noun n:

  s The pronoun '*p*' refers to <MASK>.

  s In the previous sentence, the pronoun '*p*'refers to <MASK>.

  s In the passage above, what does the pronoun'*p*' refer to? Answer: <MASK>.

  With the identity function as a verbalizer.

B Influence of the reporting method over runs

We chose to report the **accumulated maximum** performance across runs for every model. This means that if the maximum performance over random seeds is smaller than a maximum previously attained with less data points, we use the previous value. This appendix presents results with the **maximum** and **mean** at every point to condense several runs instead. Using either maximum is equivalent; using the mean, however, can make results vary significantly, as the distribution of outcomes is heavily left-skewed, or even bimodal, with poor-performance outliers.

	Average Advantage (accumulated maximum reporting)							
	MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
P vs H	3506 ± 536	752 ± 46	90 ± 2	288 ± 242	384 ± 378	282 ± 34	-424 ± 74	281 ± 137
P vs N	150 ± 252	299 ± 81	78 ± 2	-	74 ± 56	404 ± 68	-354 ± 166	-
N vs H	3355 ± 612	453 ± 90	12 ± 1	-	309 ± 320	-122 ± 62	-70 ± 160	-

Table 2: Average prompting advantage in number of data points for MNLI & SuperGLUE tasks with accmax reporting. *P* denotes the prompt model, *H* the head model. On average across performance levels, an MNLI prompt model yields the results of an MNLI head model trained with 3500 additional data points. Confidence levels are based on a multiple random runs (see text). *N* indicates a null-verbalizer prompting task that replaces the verbalizer with a non-sensical mapping. *The comparison band of MultiRC is too small as the head baseline fails to learn beyond majority class; we use the full region for a lower-bound result.

	Average Advantage (maximum reporting)							
	MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
P vs H	3506 ± 536	737 ± 53	86 ± 1	226 ± 189	448 ± 314	218 ± 24	-133 ± 65	297 ± 448
P vs N	150 ± 252	249 ± 80	75 ± 3	-	77 ± 64	331 ± 64	-341 ± 187	-
N vs H	3355 ± 612	488 ± 90	12 ± 2	-	371 ± 305	-113 ± 61	209 ± 176	-

Table 3: Average prompting advantage in number of data points for MNLI & SuperGLUE tasks with maximum reporting.

	Average Advantage (mean reporting)							
	MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
P vs H	4526 ± 266	1139 ± 285	81 ± 2	292 ± 212	399 ± 77	72 ± 71	447 ± 190	-490 ± 402
P vs N	185 ± 307	514 ± 287	80 ± 2	-	58 ± 2	621 ± 215	10 ± 104	-
N vs H	4341 ± 347	625 ± 402	1 ± 2	-	342 ± 78	-549 ± 203	437 ± 189	-

Table 4: Average prompting advantage in number of data points for MNLI & SuperGLUE tasks with mean reporting.

C Curves on all tasks

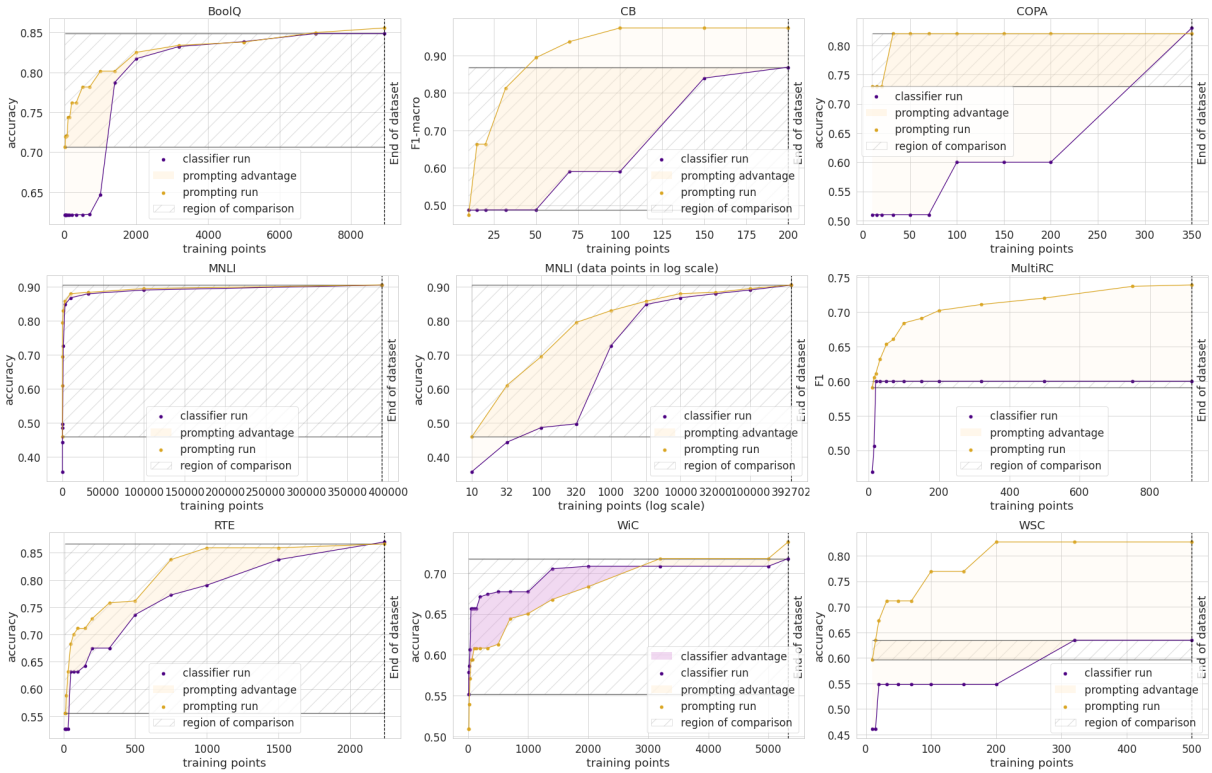


Figure 4: Prompting vs head (classifier) performance across data scales, up to the full dataset, for seven SuperGLUE tasks & MNLI. Compares the best prompt and head performance at each level of training data across 4 runs. Highlighted region shows the accuracy difference of the models. Cross-hatch region highlights the lowest- and highest- accuracy matched region in the curves. The highlighted area in this region is used to estimate the data advantage from prompting.

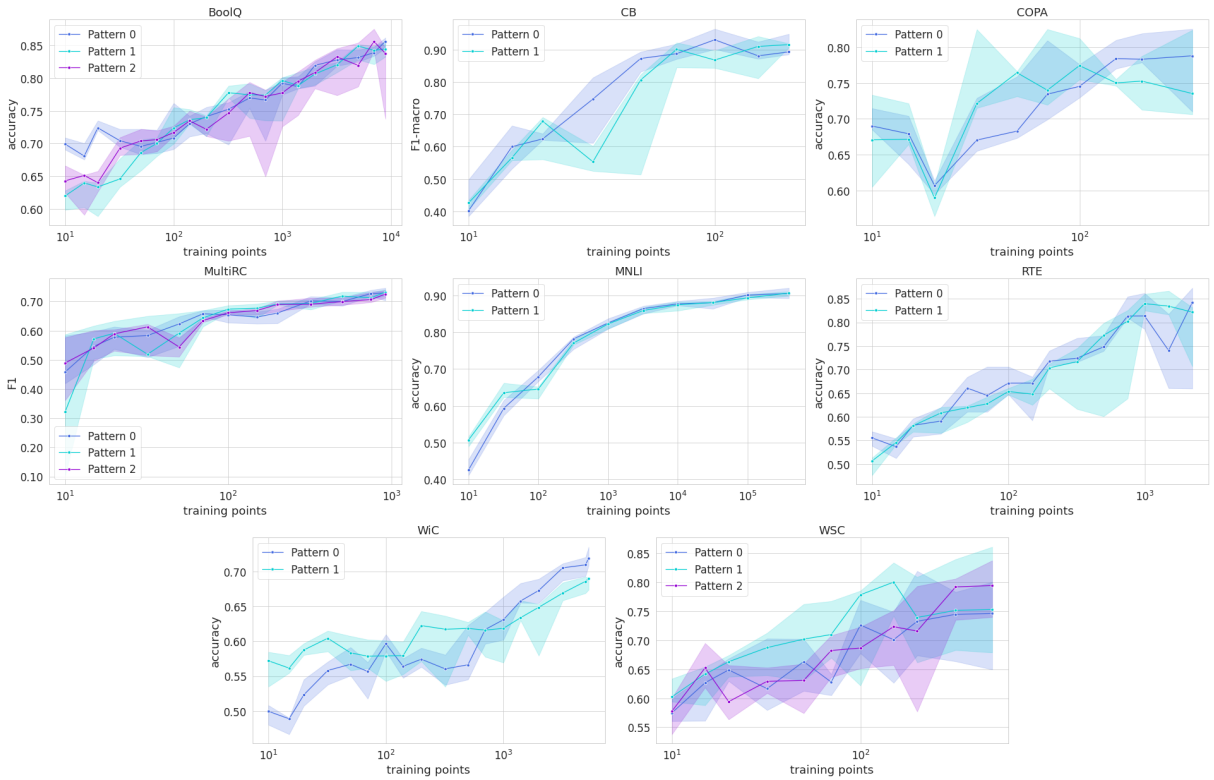


Figure 5: Median performance across runs for each prompt on every task.

Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

Sewon Min[1,2] **Xinxi Lyu**[1] **Ari Holtzman**[1] **Mikel Artetxe**[2]
Mike Lewis[2] **Hannaneh Hajishirzi**[1,3] **Luke Zettlemoyer**[1,2]
[1]University of Washington [2]Meta AI [3]Allen Institute for AI
{sewon,alrope,ahai,hannaneh,lsz}@cs.washington.edu
{artetxe,mikelewis}@meta.com

Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choce tasks, consistently over 12 different models including GPT-3. Instead, we find that other aspects of the demonstrations are the key drivers of end task performance, including the fact that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence. Together, our analysis provides a new way of understanding how and why in-context learning works, while opening up new questions about how much can be learned from large language models through inference alone.

1 Introduction

Large language models (LMs) have shown impressive performance on downstream tasks by simply conditioning on a few input-label pairs (demonstrations); this type of inference has been referred to as *in-context learning* (Brown et al., 2020). Despite in-context learning consistently outperforming zero-shot inference on a wide range of tasks (Zhao et al., 2021; Liu et al., 2021), there is little understanding of *how* it works and *which* aspects of the demonstrations contribute to end task performance.

In this paper, we show that ground truth demonstrations are in fact not required for effective in-context learning (Section 4). Specifically, replacing the labels in demonstrations with random labels barely hurts performance in a range of classification and multi-choice tasks (Figure 1). The result

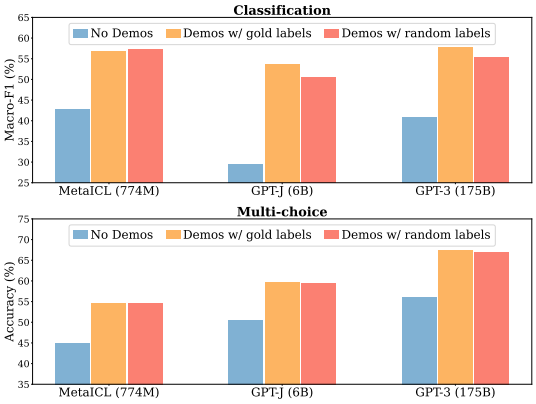


Figure 1: Results in classification (top) and multi-choice tasks (bottom), using three LMs with varying size. Reported on six datasets on which GPT-3 is evaluated; the channel method is used. See Section 4 for the full results. In-context learning performance drops only marginally when labels in the demonstrations are replaced by random labels.

is consistent over 12 different models including the GPT-3 family (Radford et al., 2019; Min et al., 2021b; Wang and Komatsuzaki, 2021; Artetxe et al., 2021; Brown et al., 2020). This strongly suggests, counter-intuitively, that the model *does not* rely on the input-label mapping in the demonstrations to perform the task.

Further analysis investigates which parts of demonstrations actually *do* contribute to the performance. We identify possible aspects of demonstrations (e.g., the label space and the distribution of the input text) and evaluate a series of variants of the demonstrations to quantify the impact of each (Section 5). We find that: (1) the label space and the distribution of the input text *specified by* the demonstrations are both key to in-context learning (regardless of whether the labels are correct for individual inputs); (2) specifying the overall format is also crucial, e.g., when the label space is unknown, using random English words as labels is significantly better than using no labels; and

(3) meta-training with an in-context learning objective (Min et al., 2021b) magnifies these effects—the models almost exclusively exploit simpler aspects of the demonstrations like the format rather than the input-label mapping.

In summary, our analysis provides a new way of understanding the role of the demonstrations in in-context learning. We empirically show that the model (1) counter-intuitively does not rely on the ground truth input-label mapping provided in the demonstrations as much as we thought (Section 4), and (2) nonetheless still benefits from knowing the label space and the distribution of inputs specified by the demonstrations (Section 5). We also include a discussion of broader implications, e.g., what we can say about the model *learning at test time*, and avenues for future work (Section 6).

2 Related Work

Large language models have been key to strong performance in a wide range of downstream tasks (Devlin et al., 2019; Radford et al., 2019; Liu et al., 2019; Raffel et al., 2020; Lewis et al., 2020). While finetuning has been a popular approach to transfer to new tasks (Devlin et al., 2019), it is often impractical to finetune a very large model (e.g. $\geq$10B parameters). Brown et al. (2020) propose in-context learning as an alternative way to learn a new task. As depicted in Figure 2, the LM learns a new task via inference alone by conditioning on a concatenation of the training data as demonstrations, without any gradient updates.

In-context learning has been the focus of significant study since its introduction. Prior work proposes better ways of formulating the problem (Zhao et al., 2021; Holtzman et al., 2021; Min et al., 2021a), better ways of choosing labeled examples for the demonstrations (Liu et al., 2021; Lu et al., 2021; Rubin et al., 2021), meta-training with an explicit in-context learning objective (Chen et al., 2021; Min et al., 2021b), and learning to follow instructions as a variant of in-context learning (Mishra et al., 2021b; Efrat and Levy, 2020; Wei et al., 2022a; Sanh et al., 2022). At the same time, some work reports brittleness and oversensitivity for in-context learning (Lu et al., 2021; Zhao et al., 2021; Mishra et al., 2021a).

Relatively less work has been done to understand why in-context learning works. Xie et al. (2022) provide theoretical analysis that in-context learning can be formalized as Bayesian inference that uses the demonstrations to recover latent concepts. Razeghi et al. (2022) show that in-context learning performance is highly correlated with term frequencies in the pretraining data. To the best of our knowledge, this paper is the first that provides an empirical analysis that investigates why in-context learning achieves performance gains over zero-shot inference. We find that the ground truth input-label mapping in the demonstrations has only a marginal effect, and measure the impact of finer-grained aspects of the demonstrations.

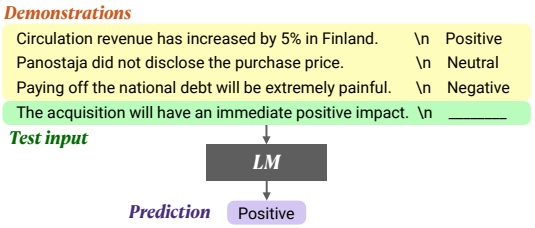


Figure 2: An overview of in-context learning. The demonstrations consist of k input-label pairs from the training data ($k = 3$ in the figure).

Model	# Params	Public	Meta-trained
GPT-2 Large	774M	✓	✗
MetaICL	774M	✓	✓
GPT-J	6B	✓	✗
fairseq 6.7B†	6.7B	✓	✗
fairseq 13B†	13B	✓	✗
GPT-3	175B‡	✗	✗

Table 1: A list of LMs used in the experiments: GPT-2 (Radford et al., 2019), MetaICL (Min et al., 2021b), GPT-J (Wang and Komatsuzaki, 2021), fairseq LMs (Artetxe et al., 2021) and GPT-3 (Brown et al., 2020). 'Public' indicates whether the model weights are public; 'Meta-trained' indicates whether the model is meta-trained with an in-context learning objective. †We use dense models in Artetxe et al. (2021) and refer them as fairseq LMs for convenience. ‡We use the Davinci API (the *base* version, not the *instruct* version) and assume it to be 175B, following Gao et al. (2021) and Artetxe et al. (2021).

3 Experimental Setup

We describe the experimental setup used in our analysis (Section 4 and 5).

Models. We experiment with 12 models in total. We include 6 language models (Table 1), all of which are decoder-only, dense LMs. We use each LM with two inference methods, direct and channel, following Min et al. (2021a). The sizes of LMs vary from 774M to 175B. We include the

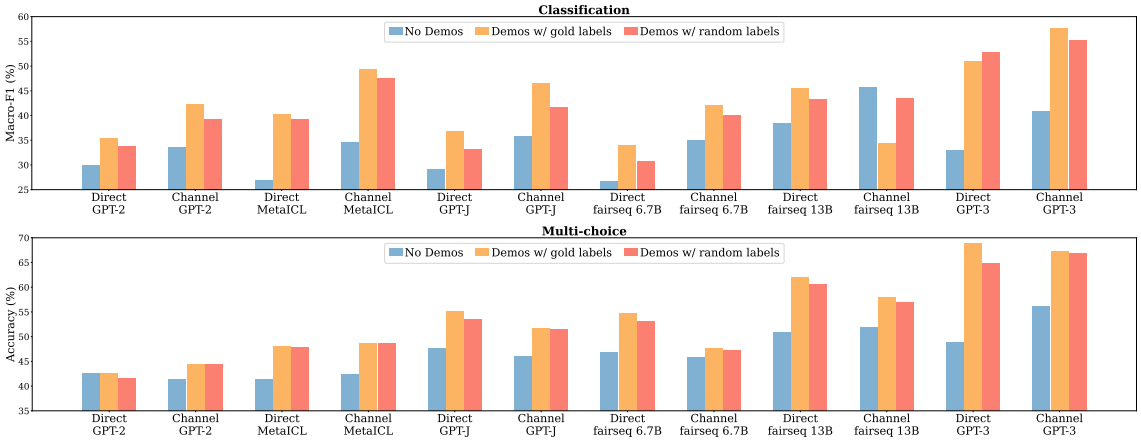


Figure 3: Results when using no-demonstrations, demonstrations with gold labels, and demonstrations with random labels in classification (top) and multi-choice tasks (bottom). The first eight models are evaluated on 16 classification and 10 multi-choice datasets, and the last four models are evaluated on 3 classification and 3 multi-choice datasets. See Figure 11 for numbers comparable across all models. **Model performance with random labels is very close to performance with gold labels** (more discussion in Section 4.1).

largest dense LM (GPT-3) and the largest publicly released dense LM (fairseq 13B) at the time of conducting experiments. We also include MetaICL, which is initialized from GPT-2 Large and then meta-trained on a collection of supervised datasets with an in-context learning objective, and ensure that our evaluation datasets do not overlap with those used at meta-training time.

Evaluation Data. We evaluate on 26 datasets, including sentiment analysis, paraphrase detection, natural language inference, hate speech detection, question answering, and sentence completion (full list and references provided in Appendix A).[1] All datasets are classification and multi-choice tasks.

We use these datasets because they (1) are true low-resource datasets with less than 10K training examples, (2) include well-studied benchmarks from GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019a), and (3) cover diverse domains including science, social media, finance, and more.

Other Details. We use $k = 16$ examples as demonstrations by default for all experiments in the paper, unless otherwise specified. Examples are sampled at uniform from the training data. We choose a set of k training examples using 5 different random seeds and run experiments 5 times. For fairseq 13B and GPT-3, due to limited resources, we experiment with a subset of 6 datasets[2] and 3 random seeds. We report Macro-F1[3] for classification tasks and Accuracy for multi-choice tasks. We compute per-dataset average over seeds, and then report macro-average over datasets. We use the minimal templates in forming an input sequence from an example. We refer to Appendix B for more details. All experiments are reproducible from `github.com/Alrope123/rethinking-demonstrations`.

4 Ground Truth Matters Little

4.1 Gold labels vs. random labels

To see the impact of correctly-paired inputs and labels in the demonstrations—which we call the ground truth input-label mapping—we compare the following three methods.[4]

No demonstrations is a typical zero-shot method that does not use any labeled data. A prediction is made via $\text{argmax}_{y \in \mathcal{C}} P(y|x)$, where x is the test input and $\mathcal{C}$ is a small discrete set of possible labels.

Demonstrations w/ gold labels are used in a typical in-context learning method with k labeled examples $(x_1, y_1)...(x_k, y_k)$. A concatenation of k input-label pairs is used to make a prediction via $\text{argmax}_{y \in \mathcal{C}} P(y|x_1, y_1 ... x_k, y_k, x)$.

[1] For convenience, we use 'labels' to refer to the output for the task, though our datasets include non-classification tasks.

[2] Three classification and three multi-choice: MRPC, RTE, Tweet_eval-hate, OpenbookQA, CommonsenseQA, COPA.

[3] Known to be better for imbalanced classes.

[4] Without loss of generality, all methods in Section 4 and 5 are described based on the direct method, but can be trivially converted to the channel method by flipping x and y.

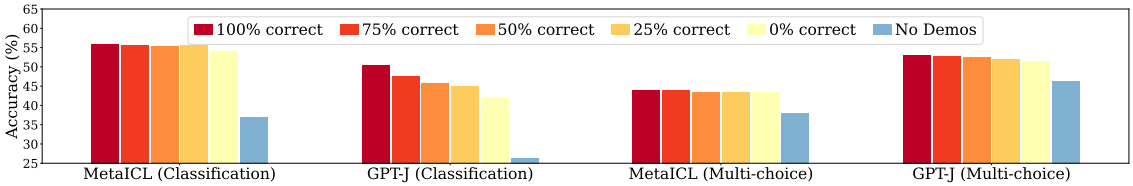


Figure 4: Results with varying number of correct labels in the demonstrations. Channel and Direct used for classification and multi-choice, respectively. Performance with no demonstrations (blue) is reported as a reference.

Demonstrations w/ random labels are formed with random labels, instead of gold labels from the labeled data. Each x_i $(1 \leq i \leq k)$ is paired with $\tilde{y}_i$ that is randomly sampled at uniform from $\mathcal{C}$. A concatenation of $(x_1, \tilde{y}_1)...(x_k, \tilde{y}_k)$ is then used to make a prediction via $\text{argmax}_{y \in \mathcal{C}} P(y|x_1, \tilde{y}_1 ... x_k, \tilde{y}_k, x)$.

Results are reported in Figure 3. First, using the demonstrations with gold labels significantly improves the performance over no demonstrations,[5] as it has been consistently found in much of prior work (Brown et al., 2020; Zhao et al., 2021; Liu et al., 2021). We then find that **replacing gold labels with random labels only marginally hurts performance**. The trend is consistent over nearly all models: models see performance drop in the range of 0–5% absolute. There is less impact in replacing labels in multi-choice tasks (1.7% on average) than in classification tasks (2.6% absolute).

This result indicates that the ground truth input-label pairs are not necessary to achieve performance gains. This is counter-intuitive, given that correctly paired training data is critical in typical supervised training—it informs the model of the expected input-label *correspondence* required to perform the downstream task. Nonetheless, the models *do* achieve non-trivial performance on the downstream tasks. This strongly suggests that the models are capable of recovering the expected input-label correspondence for the task; however, it is *not* directly from the pairings in the demonstrations.

It is also worth noting that there is particularly little performance drop in MetaICL: 0.1–0.9% absolute. This suggests that meta-training with an explicit in-context learning objective actually encourages the model to essentially ignore the input-label mapping and exploit other components of the demonstrations (more discussion in Section 5.4).

[5]There are some exceptions, e.g., in the classification tasks, Direct GPT-2, Direct GPT-J and Direct fairseq 6.7B models are not significantly better than random guessing on many datasets; Channel fairseq 13B has significantly better no-demonstrations performance compared to demonstrations with gold labels. We thus discuss the results from these models less significantly for the rest of analysis.

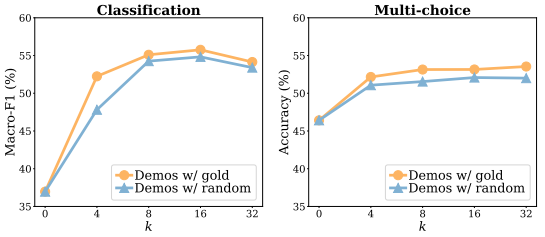


Figure 5: Ablations on varying numbers of examples in the demonstrations (k). Models that are the best under 13B in each task category (Channel MetaICL and Direct GPT-J, respectively) are used.

In Appendix C.2, we provide additional results showing that (1) selecting random labels from a true distribution of labels (instead of a uniform distribution) reduces the gap even further, and (2) the trends may depend on the dataset, although the overall trend is consistent over most datasets.

4.2 Ablations

For additional ablations, we experiment with 5 classification and 4 multi-choice datasets.[6]

Does the number of correct labels matter? To further examine the impact of correctness of labels in the demonstrations, we conduct an ablation study by varying the number of correct labels in the demonstrations. We evaluate "Demonstrations w/ a% correct labels" $(0 \leq a \leq 100)$ which consist of $k \times a/100$ correct pairs and $k \times (1 - a/100)$ incorrect pairs (see Algorithm 1 in Appendix B). Here, $a = 100$ is the same as typical in-context learning, i.e., demonstrations w/ gold labels.

Results are reported in Figure 4. Model performance is fairly insensitive to the number of correct labels in the demonstrations. In fact, always using incorrect labels significantly outperforms no-

[6]Classification includes: MRPC, RTE, Tweet_eval-hate, SICK, poem-sentiment; Multi-choice includes OpenbookQA, CommonsenseQA, COPA and ARC.

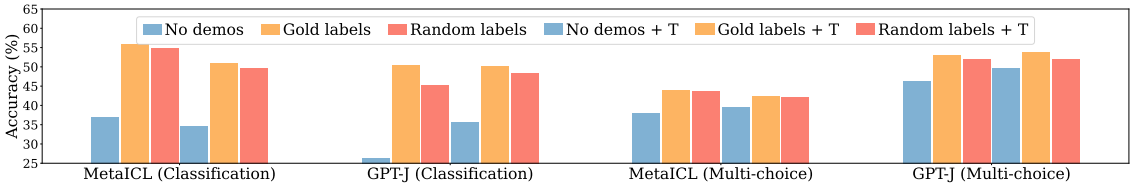


Figure 6: Results with minimal templates and manual templates. '+T' indicates that manual templates are used. Channel and Direct used for classification and multi-choice, respectively.

demonstrations, e.g., preserving 92%, 100% and 97% of improvements from using the demonstrations with MetaICL in classification, MetaICL in multi-choice, and GPT-J in multi-choice, respectively. In contrast, GPT-J in classification sees relatively significant performance drop with more incorrect labels, e.g., nearly 10% drop in performance when always using incorrect labels. Still, always using incorrect labels is significantly better than no demonstrations.

Is the result consistent with varying *k*? We study the impact of the number of input-label pairs (k) in the demonstrations. Results are reported in Figure 5. First, using the demonstrations significantly outperforms the no demonstrations method even with small k ($k = 4$), and performance drop from using gold labels to using random labels is consistently small across varying k, in the range of 0.8–1.6%.[7] Interestingly, model performance does not increase much as k increases when $k \geq 8$, both with gold labels and with random labels. This is in contrast with typical supervised training where model performance rapidly increases as k increases, especially when k is small. We hypothesize that larger labeled data is beneficial mainly for supervising the input-label correspondence, and other components of the data like the example inputs, example labels and the data format are easier to recover from the small data, which is potentially a reason for minimal performance gains from larger k (more discussion in Section 5).

Is the result consistent with better templates? While we use minimal templates by default, we also explore manual templates, i.e., templates that are manually written in a dataset-specific manner, taken from prior work (details in Appendix B). Figure 6 shows that the trend—replacing gold labels with random labels barely hurting performance—holds with manual templates. It is worth noting that using manual templates does not always outperform using minimal templates.

[7] With an exception of 4.4% in classification with $k = 4$, likely due to a high variance with a very small value of k.

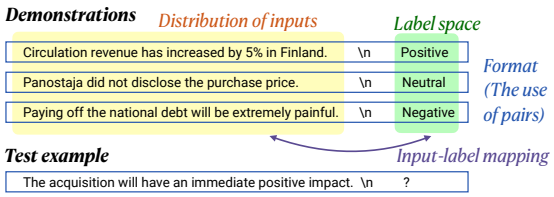


Figure 7: Four different aspects in the demonstrations: the input-label mapping, the distribution of the input text, the label space, and the use of input-label pairing as the format of the demonstrations.

5 Why *does* In-Context Learning work?

Section 4 shows that the ground truth input-label mapping in the demonstrations has little impact to performance gains from in-context learning. This section further examines what other aspects of the demonstrations lead to good performance of in-context learning.

We identify four aspects of the demonstrations $(x_1, y_1)...(x_k, y_k)$ that potentially provide learning signal (depicted in Figure 7).

1. **The input-label mapping**, i.e., whether each input x_i is paired with a correct label y_i.
2. **The distribution of the input text**, i.e., the underlying distribution that $x_1...x_k$ are from.
3. **The label space**, i.e., the space covered by $y_1...y_k$.
4. **The format**—specifically, the use of input-label pairing as the format.

As Section 4 does for the input-label mapping, we design a series of variants of the demonstrations that quantify the impact of each aspect in isolation (Section 5.1–5.3). We then additionally discuss the trend of the models meta-trained with an in-context learning objective (Section 5.4). For all experiments, models are evaluated on five classification

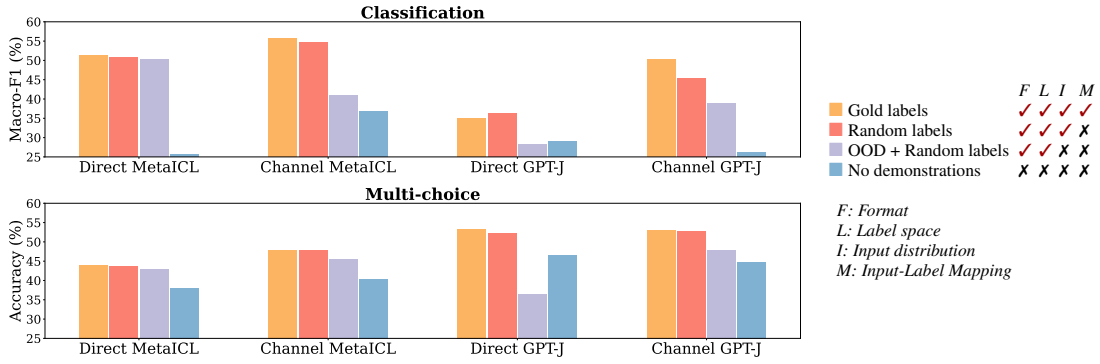


Figure 8: Impact of the distribution of the inputs. Evaluated in classification (top) and multi-choice (bottom). The impact of the distribution of the input text can be measured by comparing ■ and ■. The gap is substantial, with an exception in Direct MetaICL (discussion in Section 5.1).

and four multi-choice datasets as in Section 4.2. See Appendix B and Table 4 for implementation details and example demonstrations, respectively.

5.1 Impact of the distribution of the input text

We experiment with **OOD demonstrations** which include out-of-distribution (OOD) text instead of the inputs from unlabeled training data. Specifically, a set of k sentences $\{x_{i,\text{rand}}\}_{i=1}^{k}$ are randomly sampled from an external corpus, and replace $x_1...x_k$ in the demonstrations. This variant assesses the impact of the distribution of the input text, while keeping the label space and the format of the demonstrations.

Results. Figure 8 shows that using out-of-distribution inputs instead of the inputs from the training data significantly drops the performance when Channel MetaICL, Direct GPT-J or Channel GPT-J are used, both in classification and multi-choice, by 3–16% in absolute. In the case of Direct GPT-J in multi-choice, it is even significantly worse than no demonstrations. Direct MetaICL is an exception, which we think is the effect of meta-training (discussion in Section 5.4).

This suggests that in-distribution inputs in the demonstrations substantially contribute to performance gains. This is likely because conditioning on the in-distribution text makes the task closer to language modeling, since the LM always conditioned on the in-distribution text during training.

5.2 Impact of the label space

We also experiment with **demonstrations w/ random English words** that use random English words as labels for all k pairs. Specifically, we sample a random subset of English words $\mathcal{C}_{\text{rand}}$ where $|\mathcal{C}_{\text{rand}}| = |\mathcal{C}|$, and randomly pair $\tilde{y}_i \in \mathcal{C}_{\text{rand}}$ with x_i. This variant assesses the impact of the label space, while keeping the distribution of the input text and the format of the demonstrations.

Results. Based on Figure 9, direct models and channel models exhibit different patterns. With direct models, the performance gap between using random labels within the label space and using random English words is significant, ranging between 5–16% absolute. This indicates that conditioning on the label space significantly contributes to performance gains. This is true even for multi-choice tasks where there is no fixed set of labels—we hypothesize that multi-choice tasks still do have a particular distribution of the choices (e.g., objects like "Bolts" or "Screws" in the OpenBookQA dataset) that the model uses.

On the other hand, removing the output space does not lead to significant drop in the channel models: there is 0–2% drop in absolute, or sometimes even an increase. We hypothesize that this is because the channel models only condition on the labels, and thus are not benefiting from knowing the label space. This is in contrast to direct models which must *generate* the correct labels.

5.3 Impact of input-label pairing

Section 5.1 and 5.2 focus on variants which keep the format of the demonstrations as much as possible. This section explores variants that change the format. While there are many aspects of the format, we make minimal modifications to remove the pairings of inputs to labels. Specifically, we evaluate **demonstrations with no labels** where the LM is

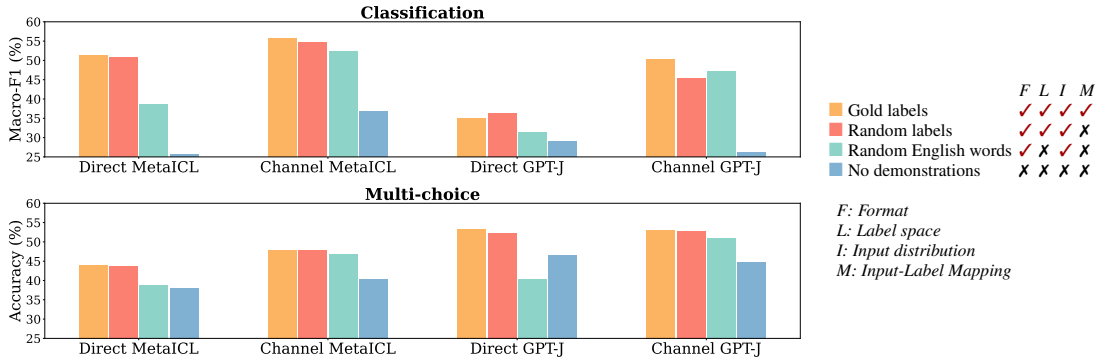


Figure 9: Impact of the label space. Evaluated in classification (top) and multi-choice (bottom). The impact of the label space can be measured by comparing ■ and ■. The gap is significant in the direct models but not in the channel models (discussion in Section 5.2).

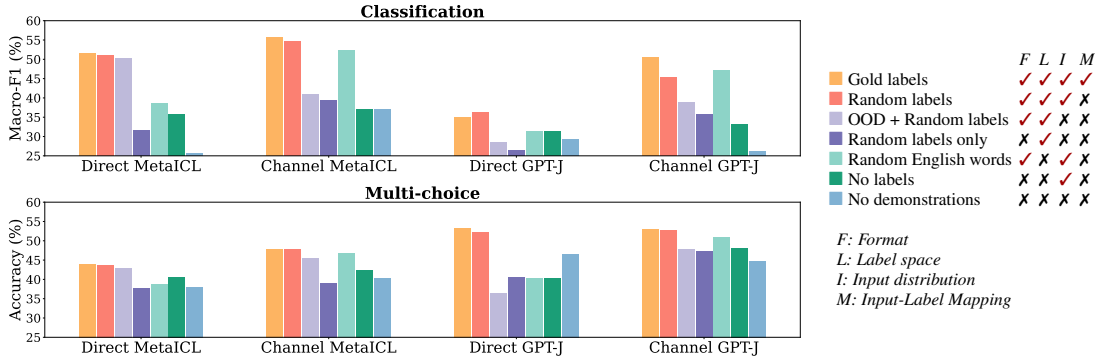


Figure 10: Impact of the format, i.e., the use of the input-label pairs. Evaluated in classification (top) and multi-choice (bottom). Variants of demonstrations without keeping the format (■ and ■) are overall not better than no demonstrations (■). Keeping the format is especially significant when it is possible to achieve substantial gains with the label space but without the inputs (■ vs. ■ in Direct MetaICL), or with the input distribution but without the labels (■ vs. ■ in Channel MetaICL and Channel GPT-J). More discussion in Section 5.3.

conditioned on the concatenation of $x_1...x_k$, and **demonstrations with labels only** where the LM is conditioned on the concatenation of $y_1...y_k$. These ablations provide the no-format counterparts of the 'demonstrations with random English words' and 'demonstrations with OOD inputs', respectively.

Results. Based on Figure 10, removing the format is close to or worse than no demonstrations, indicating the importance of the format. This is likely because conditioning on a sequence of input-label pairs triggers the model to mimic the overall format and complete the new example as expected when the test input is given.

More interestingly, keeping the format plays a significant role in retaining a large portion of performance gains by only using the inputs or only using the labels. For instance, with Direct MetaICL, it is possible to retain 95% and 82% of improvements from in-context learning (demonstrations with gold labels) by simply sampling random sentences from a corpus and randomly pairing them with the label set (■ in Figure 10) in classification and multi-choice, respectively. Similarly, with the channel models, it is possible to retain 82%, 87%, 86% and 75% of improvements from in-context learning by simply pairing each input from the unlabeled training data with a random English word (■ in Figure 10) in MetaICL classification, GPT-J classification, MetaICL multi-choice and GPT-J multi-choice, respectively. For all of these cases, removing inputs instead of using OOD inputs, or removing labels instead of using random English words is significantly worse, indicating that **keeping the format of the input-label pairs is key**.

5.4 Impact of meta-training

Different from other models, MetaICL is trained with an in-context learning objective, in line with

recent work that uses multi-task training on a large collection of supervised datasets (called meta-training) for generalization to new tasks (Aghajanyan et al., 2021; Khashabi et al., 2020; Wei et al., 2022a; Sanh et al., 2022). We aim to better understand the role of this meta-training in relation with our findings by closely examining the result of MetaICL. In particular, we observe that the patterns we see so far are significantly more evident with MetaICL than with other models. For instance, the ground truth input-label mapping matters even less, and keeping the format of the demonstrations matters even more. There is nearly zero influence of the input-label mapping and the input distribution in Direct MetaICL, and the input-label mapping and the output space in Channel MetaICL.

Based on this observation, we hypothesize that **meta-training encourages the model to exclusively exploit simpler aspects of the demonstrations and to ignore others**. This is based on our intuition that (1) the input-label mapping is likely harder to exploit, (2) the format is likely easier to exploit, and (3) the space of the text that the model is trained to generate is likely easier to exploit than the space of the text that the model conditions on.[8]

6 Discussion & Conclusion

In this paper, we study the role of the demonstrations with respect to the success of in-context learning. We find that the ground truth input-label mapping in the demonstrations matters significantly less than one might think—replacing gold labels with random labels in the demonstrations only marginally lowers the performance. We then identify a series of aspects in the demonstrations and examine which aspect actually contributes to performance gains. Results reveal that (1) gains are mainly coming from *independent* specification of the input space and the label space, (2) the models can still retain up to 95% of performance gains by using either the inputs only or the label set only if the right format is used, and (3) meta-training with an in-context learning objective magnifies these trends. Together, our findings lead to a set of broader indications about in-context learning, as well as avenues for future work.

Does the model *learn* at test time? If we take a strict definition of learning: capturing the input-label correspondence given in the training data, then our findings suggest that LMs do not learn new tasks at test time. Our analysis shows that the model may ignore the task defined by the demonstrations and instead use prior from pretraining.

However, *learning* a new task can be interpreted more broadly: it may include adapting to specific input and label distributions and the format suggested by the demonstrations, and ultimately getting to make a prediction more accurately. With this definition of learning, the model *does* learn the task from the demonstrations. Our experiments indicate that the model *does* make use of aspects of the demonstrations and achieve performance gains.

Capacity of LMs. The model performs a downstream task without relying on the input-label correspondence from the demonstrations. This suggests that the model has learned the (implicit notion of) input-label correspondence from the language modeling objective alone, e.g., associating a positive review with the word 'positive'. This is in line with Reynolds and McDonell (2021) who claim that the demonstrations are for *task location* and the intrinsic ability to perform the task is obtained at pretraining time.[9]

On one hand, this suggests that the language modeling objective has led to great zero-shot *capacity*, even if it is not always evident from the naive zero-shot *accuracy*. On the other hand, this suggests that in-context learning may not work on a task whose input-label correspondence is not already captured in the LM. This leads to the research question of how to make progress in NLP problems that in-context learning does not solve: whether we need a better way of extracting the input-label mappings that are already stored in the LM, a better variant of the LM objective that learns a wider range of task semantics, or explicit supervision through fine-tuning on the labeled data.

Connection to instruction-following models. Prior work has found it promising to train the model that reads the natural language description of the task (called instructions) and performs a new task at inference (Mishra et al., 2021b; Efrat and Levy, 2020; Wei et al., 2022a; Sanh et al., 2022). We think the demonstrations and instructions largely have the same role to LMs, and hypothesize that our

[8] That is, the direct model exploits the label space better than the input distribution, and the channel model exploits the input distribution better than the label space.

[9] However, while Reynolds and McDonell (2021) claims that the demonstrations are thus unnecessary, we think using the demonstrations is actually the most unambiguous and the easiest way to prompt the model to perform a task.

findings hold for instruction-following models: the instructions prompt the model to recover the capacity it already has, but do not supervise the model to learn novel task semantics. This has been partially verified by Webson and Pavlick (2022) who showed that the model performance does not degrade much with irrelevant or misleading instructions. We leave more analysis on instruction-following models for future work.

Significantly improved zero-shot performance. One of our key findings is that it is possible to achieve nearly k-shot performance without using any labeled data, by simply pairing each unlabeled input with a random label and using it as the demonstrations. This means our zero-shot baseline level is significantly higher than previously thought.[10] Future work can further improve the zero-shot performance with relaxed assumptions in access to the unlabeled training data.

Limitation

Effect of types of tasks and datasets. This paper focuses on the tasks from established NLP benchmarks that have *real* natural language inputs. Synthetic tasks with more limited inputs may actually use the ground truth labels more, as observed by Rong (2021).

We report macro-level analysis by examining the average performance over multiple NLP datasets, but different datasets may behave differently. Appendix C.2 discusses this aspect, including findings that there are larger gaps between using the ground truth labels and using the random labels in some dataset-model pairs (e.g., in the most extreme case, nearly 14% absolute on the financial_phrasebank dataset with GPT-J). Since the first version of our paper, Kim et al. (2022) showed that using negated labels substantially lowers the performance in classification.[11] We believe it is important to understand to what extend the model needs the ground truth labels to successfully perform in-context learning.

Extensions to generation. Our experiments are limited to classification and multi-choice tasks. We hypothesize that ground truth output may not be necessary for in-context learning in the open-set tasks such as generation, but leave this to future work. Extending of our experiments to such tasks is not trivial, because it requires a variation of the output which has incorrect input-output correspondence while keeping the correct output distribution (which is important based on our analysis in Section 5).

Since the first version of our paper, Madaan and Yazdanbakhsh (2022) conducted a similar analysis with the chain of thought prompting (Wei et al., 2022b) which generates a rationale to perform complex tasks such as math problems. Madaan and Yazdanbakhsh (2022) show that, while simply using a random rationale in the demonstrations (e.g., pairing with a rationale from a different example) significantly degrades the performance, other types of counterfactual rationales (e.g., wrong equations) do not degrade the performance as much as we thought. We refer to Madaan and Yazdanbakhsh (2022) for more discussions on what aspects of the rationale matter or do not matter.

[10] We take the perspective that using the unlabeled training data is permitted (Kodirov et al., 2015; Wang et al., 2019b; Schick and Schütze, 2021).

[11] Note that Kim et al. (2022) estimate the random label performance by interpolating with the performance using negated labels, while our paper samples the random labels at uniform.

Acknowledgements

We thank Gabriel Ilharco, Julian Michael, Ofir Press, UW NLP members and anonymous reviewers for their comments in the paper. This research was supported by NSF IIS-2044660, ONR N00014-18-1-2826, a Sloan fellowship and gifts from AI2.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*.

Mikel Artetxe, Shruti Bhosale, Naman Goyal, Todor Mihaylov, Myle Ott, Sam Shleifer, Xi Victoria Lin, Jingfei Du, Srinivasan Iyer, Ramakanth Pasunuru, et al. 2021. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of EMNLP*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *NeurIPS*.

Michael Chen, Mike D'Arcy, Alisa Liu, Jared Fernandez, and Doug Downey. 2019. CODAH: An adversarially-authored question answering dataset for common sense. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2021. Meta-learning via language model in-context tuning. *arXiv preprint arXiv:2110.07814*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*.

Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. 2018. Hate Speech Dataset from a White Supremacy Forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

T. Diggelmann, Jordan L. Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2020. Climate-fever: A dataset for verification of real-world climate claims. *ArXiv*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

L Gao, S Biderman, S Black, L Golding, T Hoppe, C Foster, J Phang, H He, A Thite, N Nabeshima, et al. 2021. The pile: an 800gb dataset of diverse text for language modeling 2020. *arXiv preprint arXiv:2101.00027*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *The First Joint Conference on Lexical and Computational Semantics (SemEval)*.

Ari Holtzman, Peter West, Vered Schwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *EMNLP*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single qa system. In *Findings of EMNLP*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI*.

Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, Kang Min Yoo, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. *arXiv preprint arXiv:2205.12685*.

Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. 2015. Unsupervised domain adaptation for zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid,

Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *EMNLP: System Demonstrations*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Robert L Logan IV, Ivana Balaževic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*.

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. *arXiv preprint arXiv:2104.08786*.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *J. Assoc. Inf. Sci. Technol.*

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *LREC*.

Clara H. McCreery, Namit Katariya, Anitha Kannan, Manish Chablani, and Xavier Amatriain. 2020. Effective transfer learning for identifying similar questions: Matching user questions to covid-19 faqs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021a. Noisy channel language model prompting for few-shot text classification. *arXiv preprint arXiv:2108.04106*.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021b. MetaICL: Learning to learn in context. *arXiv preprint*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2021a. Reframing instructional prompts to gptk's language. *arXiv preprint arXiv:2109.07830*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021b. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2020. Ethos: an online hate speech detection dataset. *ArXiv*.

Sebastian Nagel. 2016. CC-News. http://web.archive.org/save/http://commoncrawl.org/2016/10/news-dataset-available.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning. *arXiv preprint arXiv:2202.07206*.

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*.

Frieda Rong. 2021. Extrapolating to unnatural language processing with gpt-3's in-context learning: The good, the bad, and the mysterious. https://ai.stanford.edu/blog/in-context-learning.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *NAACL-HLT*.

Emily Sheng and David Uthus. 2020. Investigating societal biases in a poetry composition system. In *Proceedings of the Second Workshop on Gender Bias in Natural Language Processing*.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge data set and models for dialogue-based reading comprehension. *TACL*.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019a. Quarel: A dataset and models for answering questions about qualitative relationships. In *AAAI*.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019b. QuaRTz: An open-domain dataset of qualitative relationship questions. In *EMNLP*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL-HLT*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. 2019b. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *NAACL-HLT*.

Jason Wei, Maarten Bosma, Vincent Y 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 *ICLR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *ICLR*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. In *EMNLP*.

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

A Full Datasets

We include 26 datasets as follows: financial_phrasebank (Malo et al., 2014), poem_sentiment (Sheng and Uthus, 2020), medical_questions_pairs (McCreery et al., 2020), glue-mrpc (Dolan and Brockett, 2005), glue-wnli (Levesque et al., 2012), climate_fever (Diggelmann et al., 2020), glue-rte (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), superglue-cb (de Marneffe et al., 2019), sick (Marelli et al., 2014) , hate_speech18 (de Gibert et al., 2018), ethos-national_origin (Mollas et al., 2020), ethos-race (Mollas et al., 2020), ethos-religion (Mollas et al., 2020), tweet_eval-hate (Barbieri et al., 2020), tweet_eval-stance_atheism (Barbieri et al., 2020), tweet_eval-stance_feminist (Barbieri et al., 2020), quarel (Tafjord et al., 2019a), openbookqa (Mihaylov et al., 2018), qasc (Khot et al., 2020), commonsense_qa (Talmor et al., 2019), ai2_arc (Clark et al., 2018), codah (Chen et al., 2019), superglue-copa (Gordon et al., 2012), dream (Sun et al., 2019), quartz-with_knowledge (Tafjord et al., 2019b), quartz-no_knowledge (Tafjord et al., 2019b). The choice of datasets is made following low-resource datasets in Min et al. (2021b), with the exact same set of k-shot train data using 5 random seeds. We use the HuggingFace version of the data (Lhoest et al., 2021) and use the development data for evaluation, following Ye et al. (2021). See Table 2 for statistics.

Dataset	# Train	# Eval
Task category: Sentiment analysis		
financial_phrasebank	1,811	453
poem_sentiment	892	105
Task category: Paraphrase detection		
medical_questions_pairs	2,438	610
glue-mrpc	3,668	408
Task category: Natural language inference		
glue-wnli	635	71
climate_fever	1,228	307
glue-rte	2,490	277
superglue-cb	250	56
sick	4,439	495
Task category: Hate speech detection		
hate_speech18	8,562	2,141
ethos-national_origin	346	87
ethos-race	346	87
ethos-religion	346	87
tweet_eval-hate	8,993	999
tweet_eval-stance_atheism	461	52
tweet_eval-stance_feminist	597	67
Task category: Question answering		
quarel	1,941	278
openbookqa	4,957	500
qasc	8,134	926
commonsense_qa	9,741	1,221
ai2_arc	1,119	299
Task category: Sentence completion		
codah	1665	556
superglue-copa	400	100
dream	6116	2040
quartz-with_knowledge	2696	384
quartz-no_knowledge	2696	384

Table 2: 26 datasets used for experiments, classified into 6 task categories. # Train and # Test indicate the number of training and test examples of the dataset. Note that # train is based on the original training dataset but we use k random samples for k-shot evaluation.

B Experimental Details

Example template We follow Ye et al. (2021); Min et al. (2021b); Logan IV et al. (2021) in using the minimal format to transform the input to a sequence (e.g. a concatenation of multiple inputs) and using the label words from each dataset as it is. We also explore manual templates taken from prior work (Holtzman et al., 2021; Zhao et al., 2021) as reported in Section 4.2, although we find that using these templates is not consistently better than using minimal templates. We thus run main experiments with minimal templates. Example templates are provided in Table 3.

Format of the demonstrations We follow the standard of each model for formatting the demonstrations, either from exploration in prior work or the example code provided in the official tutorial. For GPT-2, we separate the input and the label, and each demonstration example with a space. For MetaICL, GPT-J and GPT-3, we separate the input and the label with a newline (`\n`), and each demonstration example with three newlines. For fairseq models, we use a newline to separate the input and the label as well as each demonstration example.

Details in variants of the demonstrations For "demonstrations w/ $a\%$ accurate labels" ($0 \leq a \leq 100$), we use $k \times a/100$ correct pairs and $k \times (1 - a/100)$ incorrect pairs in a random order, as described in Algorithm 1. For "OOD demonstrations", we use CC-News (Nagel, 2016) as an external corpus. We consider the length of the text during sampling, so that sampled sentences have similar length to the test input. For "demonstrations with random English words", we use `pypi.org/project/english-words` for the set of En-

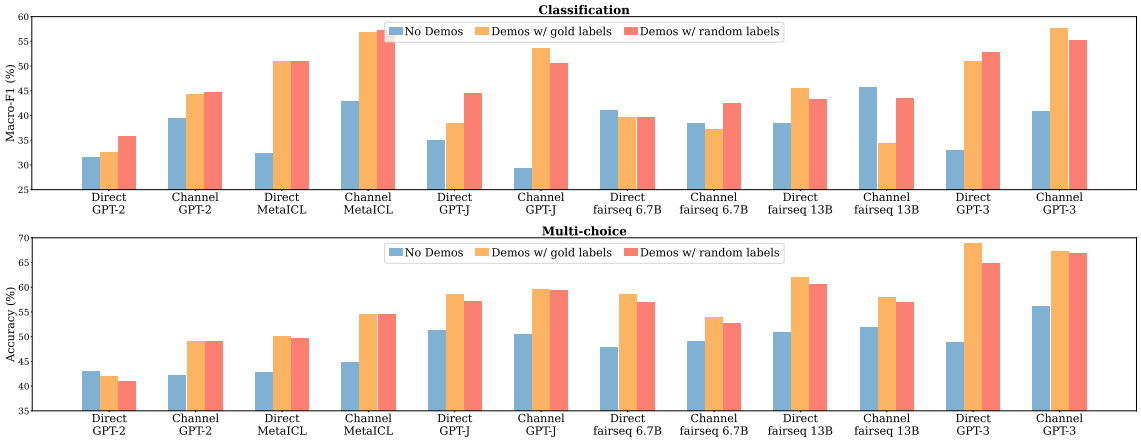


Figure 11: Results of No-demonstration, Gold demonstration and Random demonstration on 3 classification datasets (top) and 3 multi-choice datasets (bottom). Details in Section 4.1. This figure is for providing numbers that are comparable across models—full results with more datasets are reported in Figure 3.

Algorithm 1 Forming the demonstrations with an accuracy of $a\%$.

procedure FORMDEMONS($\{(x_i, y_i)\}_{i=1}^{k}, a$)
  $D \leftarrow []$ *// demonstration to be formed*
  $n \leftarrow k \times a/100$ *// number of correct pairs*
  $\mathcal{G} \leftarrow \text{Sample}(\text{Range}(1, k), n)$
  for $i \in \text{Range}(1, k)$ **do**
    if $i \in \mathcal{G}$ **then** *// add correct pair*
      $D.\text{append}((x_i, y_i))$
    else *// add incorrect pair*
      $D.\text{append}((x_i, \text{Sample}(\mathcal{C} - y_i)))$
  return D

glish words, which consists of 61,569 words.

Table 4 provides a list of example demonstrations for each method used in Section 5.

C More Experimental Results

C.1 Gold labels vs. random labels

Figure 11 shares the same interface as Figure 3, but all models are evaluated on 3 classification and 3 multi-choice datasets and are thus comparable to each other.

C.2 Random labels from true distribution of labels & Task breakdown

In Section 4, random labels are sampled from the label space from a uniform distribution. We experiment with another variant of demonstrations in the classification tasks, where labels are randomly sampled from the true distribution of labels on the training data. This may have large impact if labels are far from uniform on the training data. Results indicate that performance drop from using gold labels is further reduced compared to using uniformly random labels: with Channel MetaICL, the gap is reduced from 1.9% to 1.3% absolute, and with Channel GPT-J, the gap is reduced from 5.0% to 3.5% absolute.

Figure 12 shows performance gap between using gold labels and using random labels per dataset. We find that the trend that the gap is smaller than previously thought is consistant across most datasets. Nonetheless, there are a few outlier datasets where performance gap is non-negligible, such as financial_phrasebank and a few hate speech detection datasets. Future work may investigate on which tasks the model makes more use of the correctly paired training data.

C.3 More variants of the demonstrations

We explored **demonstrations with a constant label** where all labels in the demonstrations are replaced with a constant text, "`answer`". Specifically, a prediction is made via $\text{argmax}_{y \in \mathcal{C}} P(y|x_1, \texttt{answer} ... x_k, \texttt{answer}, x)$. This can be viewed as another way to remove the impact of the label space while keeping the impact of the distribution of the input text. However, results are consistently worse than the results of demonstrations with random English labels. We think this is because constant labels actually change the format of the demonstrations, since they can be viewed as part of a separator between different demonstration examples.

We also explored **demonstrations with the test input** where all inputs in the demonstrations are replaced with the test input, each paired with a ran-

dom label. Specifically, a prediction is made via $\text{argmax}_{y \in \mathcal{C}} P(y|x, \tilde{y}_1 ... x, \tilde{y}_k, x)$, where $\tilde{y}_i$ ($1 \leq i \leq k$) is randomly sampled at uniform from $\mathcal{C}$. This variant is seemingly a reasonable choice given that it satisfies the condition that the inputs in the demonstrations come from the same distribution as the test input (since they are identical), and using random labels is as good as using gold labels. Nonetheless, we find that this variant is significantly worse than most other methods with demonstrations. We think this is because using the constant input for all demonstration example significantly changes the format of the sequence, since the input can be viewed as part of a separator between different demonstration examples.

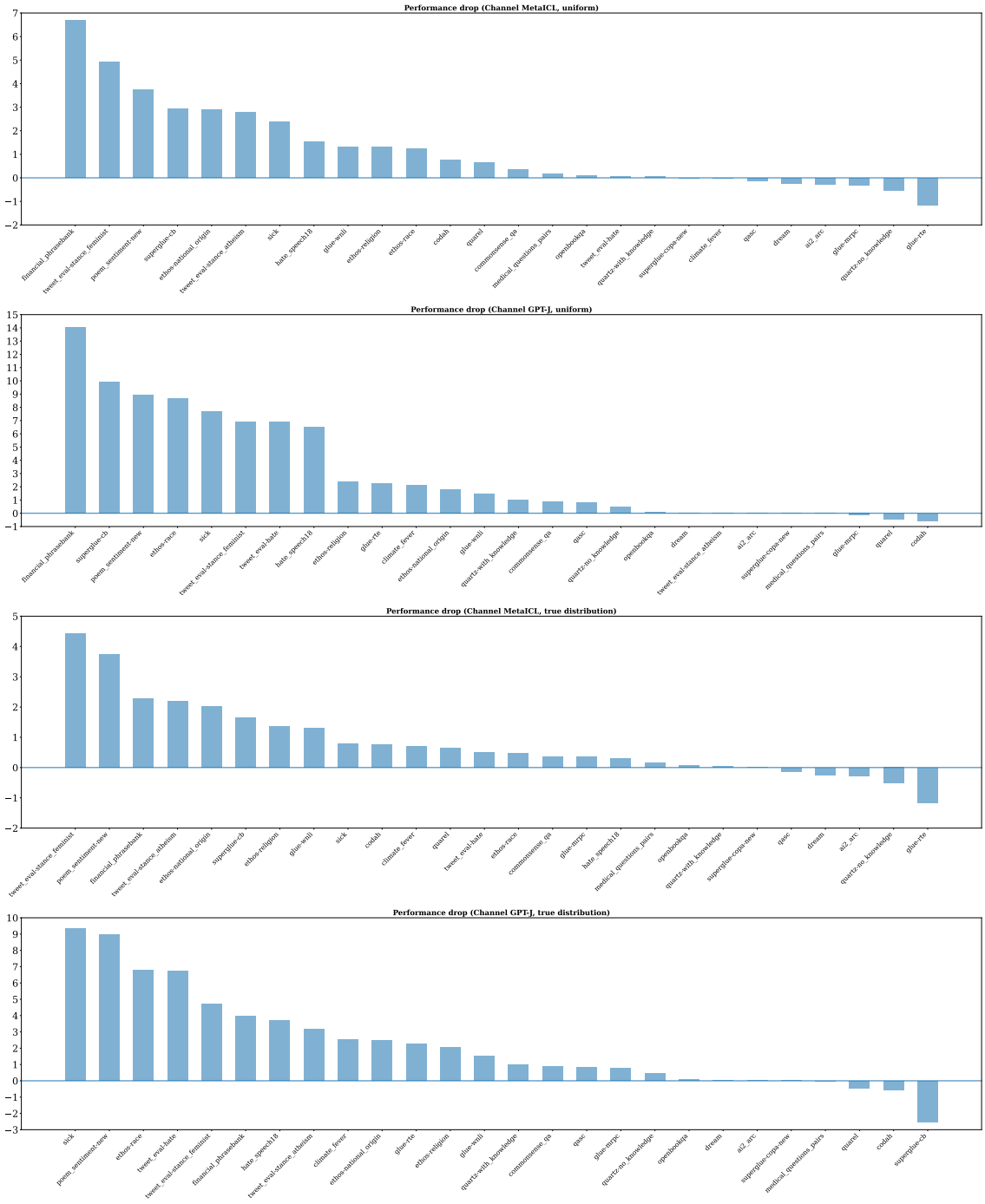


Figure 12: Performance gap from using the demonstrations with gold labels to using the demonstrations with random labels. Datasets are sorted in descending order. The top two figures use random labels that are sampled at uniform, with Channel MetaICL and Channel GPT-J, respectively. The bottom two figures use random labels that are sampled from a true distribution of labels on the training data, with Channel MetaICL and Channel GPT-J, respectively.

Dataset	Type	Example
MRPC	Minimal	sentence 1: Cisco pared spending to compensate for sluggish sales . [SEP] sentence 2: In response to sluggish sales , Cisco pared spending . \n {equivalent|not_equivalent}
	Manual	Cisco pared spending to compensate for sluggish sales . \n The question is: In response to sluggish sales , Cisco pared spending . True or False? \n The answer is:{True|False}
RTE	Minimal	sentence 1: The girl was found in Drummondville. [SEP] sentence 2: Drummondville contains the girl. \n {entailment|not_entailment}
	Manual	The girl was found in Drummondville. \n The question is: Drummondville contains the girl. True or False? \n The answer is:{True|False}
Tweet_eval-hate	Minimal	The Truth about #Immigration \n {hate|non-hate}
	Manual	Tweet: The Truth about #Immigration \n Sentiment: {against|favor}
SICK	Minimal	sentence 1: A man is screaming. [SEP] sentence 2: A man is scared. \n {contradiction|entailment|neutral}
	Manual	A man is screaming. \n The question is: A man is scared. True or False? \n The answer is: {False|True|Not sure}
poem-sentiment	Minimal	willis sneered: \n {negative|no_impact|positive}
	Manual	willis sneered: \n The sentiment is: {negative|no_impact|positive}
OpenbookQA	Minimal	What creates a valley? \n {feet|rock|water|sand}
	Manual	The question is: What creates a valley? \n The answer is: {feet|rock|water|sand}
CommonsenseQA	Minimal	What blocks sunshine? \n {summer|park|desktop|sea|moon}
	Manual	The question is: What blocks sunshine? \n The answer is: {summer|park|desktop|sea|moon}
COPA	Minimal	Effect: I coughed. \n {Cause: I inhaled smoke.|Cause: I lowered my voice.}
	Manual	I coughed because {I inhaled smoke.|I lowered my voice.}
ARC	Minimal	Which biome has the most vegetation? \n {desert|forest|grassland|tundra}
	Manual	The question is: Which biome has the most vegetation? \n The answer is: {desert|forest|grassland|tundra}

Table 3: A list of minimal templates taken from Ye et al. (2021); Min et al. (2021b) and manual templates taken from Holtzman et al. (2021); Zhao et al. (2021). Details provided in Appendix B. See Figure 6 for discussion in empirical results. The input and the label are in the red text and in the blue text, respectively. Note that | is used to separate different options for the labels.

Demos w/ gold labels	(*Format* ✓ *Input distribution* ✓ *Label space* ✓ *Input-label mapping* ✓) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n positive Panostaja did not disclose the purchase price. \n neutral
Demos w/ random labels	(*Format* ✓ *Input distribution* ✓ *Label space* ✓ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n neutral Panostaja did not disclose the purchase price. \n negative
OOD Demos w/ random labels	(*Format* ✓ *Input distribution* ✗ *Label space* ✓ *Input-label mapping* ✗) Colour-printed lithograph. Very good condition. Image size: 15 x 23 1/2 inches. \n neutral Many accompanying marketing claims of cannabis products are often well-meaning. \n negative
Demos w/ random English words	(*Format* ✓ *Input distribution* ✓ *Label space* ✗ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n unanimity Panostaja did not disclose the purchase price. \n wave
Demos w/o labels	(*Format* ✗ *Input distribution* ✓ *Label space* ✗ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. Panostaja did not disclose the purchase price.
Demos labels only	(*Format* ✗ *Input distribution* ✗ *Label space* ✓ *Input-label mapping* ✗) positive neutral

Table 4: Example demonstrations when using methods in Section 5. The financial_phrasebank dataset with $\mathcal{C} = \{$"positive", "neutral", "negative"$\}$ is used. Red text indicates the text is sampled from an external corpus; blue text indicates the labels are randomly sampled from the label set; purple text indicates a random English word.

Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

Anonymous ACL submission

We are glad that all reviewers find that the paper is novel (8jk5, LQ6N, 92YB, 7E5P), of interest to the broader NLP community (LQ6N, 92YB, 7E5P), supported by solid experiments (8jk5, LQ6N, 92YB, 7E5P), and well-written (8jk5, LQ6N, 92YB). Reviewers gave comments on more discussion, limitations and avenues for future work. We will incorporate them in the next version of the paper.[1]

Adding discussion on robustness of LMs (8jk5, 7E5P): We think the fact that LMs do not use the input-label correspondence does not necessarily mean that LMs are robust to other aspects of the demonstration. Nonetheless, it will be an interesting avenue for future work, given that LMs are highly sensitive to small changes in the demonstrations (**??**).

Adding discussion with respect to the model size (8jk5): Absolute differences are similar across different model sizes (Figure 3), but since the large models have higher absolute performance, the relative differences are larger with larger models.

When our findings hold or do not hold (8jk5): We believe that the findings will hold only when some notion of input-label correspondences are already captured during pre-training—for tasks whose input-label correspondences during pretraining is sparse, the demonstrations with random labels are highly unlikely to work. This has been shown by **?** in a synthetic setup, and future work can revisit this with more natural data.

Risk in applying in-context learning, Recommendation to practitioners (8jk5, LQ6N): Since the models do not capture the correspondence from the demonstrations, it is possible that models are simply relying on some notation of input-label correspondence during pre-training. Thus, applying in-context learning for problems that were not in the pretraining data would be potentially risky, and practitioners may want to collect the labeled data and fine-tune the model.

Where do the gains from demonstrations come from? (92YB): We think gains from demonstrations are mainly due to the specification of the input distribution and the label space rather than the input-label correspondence, as Section 5 indicates. We will clarify this in the next version of the paper.

Concrete answer to "why" using random labels keeps reasonable performance (LQ6N): We think it is highly likely to be because the model has been exposed to some notion of input-label correspondence during pre-training, so that the demonstrations play a role of triggering them at inference.

Consideration when training large LMs (LQ6N): Due to compute limitations, we were not be able to provide recommendations that are empirically supported. Future work may investigate aspects of language model pretraining that affect the findings, including the pretraining data and the objective.

Word ordering matters? (7E5P): We think it is an important avenue for future work. For this paper, we did not include it in our scope due to requiring multiple times more experiments.

More task types, e.g., generation (7E5P): Extending this work to generation tasks is not very trivial because designing the demonstrations that do not keep the input-output correspondence but keep the output distribution is difficult. For instance, if we simply replace the output with a random output as we did in the classification tasks, it will destroy both the input-output correspondence and the output distribution. We hope future work can investigate more in this direction.

Stronger link with instruction-following models (7E5P): We will add discussion on **?** who find that instructions that are irrelevant or even misleading lead to performance gains as much as "good"

[1] We answered reviewers' questions on the OpenReview page, and address higher-level comments here.

instructions do.

Limitation

This paper focuses on the tasks from established NLP benchmarks that have *real* natural language inputs. Synthetic tasks with more limited inputs may actually use the ground truth labels more, as observed by **?**. Our paper mainly includes macro-level analysis by examining the average performance over multiple NLP datasets, but different datasets may behave differently. Appendix discusses this aspect, including findings that there are larger gaps between using the ground truth labels and using the random labels in some dataset-model pairs (e.g., in the most extreme case, nearly 14% absolute on the financial_phrasebank dataset with GPT-J). We believe it is important to understand in which cases the ground truth labels matter or not, which we leave for future work. Furthermore, our work is limited to classification and multi-choice tasks. Extension to the open-set tasks such as generation is not trivial, since it is unclear how to remove the input-output correspondence while keeping the correct output distribution. We leave the extension for future work.

Large Language Models Struggle to Learn Long-Tail Knowledge

Nikhil Kandpal [1] **Haikang Deng** [1] **Adam Roberts** [2] **Eric Wallace** [3] **Colin Raffel** [1]

Abstract

The Internet contains a wealth of knowledge—from the birthdays of historical figures to tutorials on how to code—all of which may be learned by language models. However, while certain pieces of information are ubiquitous on the web, others appear extremely rarely. In this paper, we study the relationship between the knowledge memorized by large language models and the information in pre-training datasets scraped from the web. In particular, we show that a language model's ability to answer a fact-based question relates to how many documents associated with that question were seen during pre-training. We identify these relevant documents by entity linking pre-training datasets and counting documents that contain the same entities as a given question-answer pair. Our results demonstrate strong correlational and causal relationships between accuracy and relevant document count for numerous question answering datasets (e.g., TriviaQA), pre-training corpora (e.g., ROOTS), and model sizes (e.g., 176B parameters). Moreover, while larger models are better at learning long-tail knowledge, we estimate that today's models must be scaled by many orders of magnitude to reach competitive QA performance on questions with little support in the pre-training data. Finally, we show that retrieval-augmentation can reduce the dependence on relevant pre-training information, presenting a promising approach for capturing the long-tail.

1. Introduction

Large language models (LLMs) trained on text from the Internet capture many facts about the world, ranging from well-known factoids to esoteric domain-specific information. These models implicitly store knowledge in their parameters

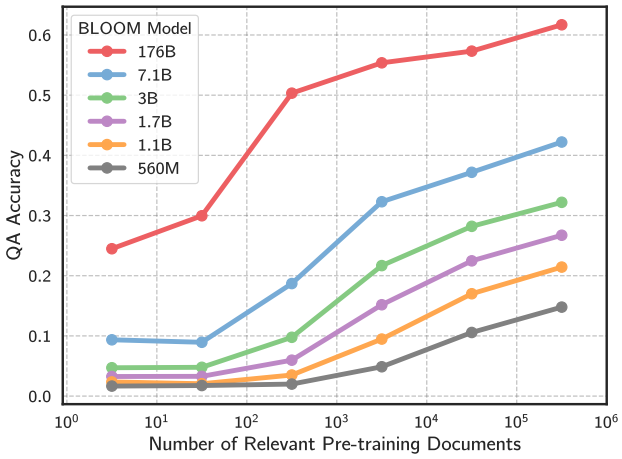


Figure 1. Language models struggle to capture the long-tail of information on the web. Above, we plot accuracy for the BLOOM model family on TriviaQA as a function of how many documents in the model's pre-training data are relevant to each question.

(Petroni et al., 2019; Roberts et al., 2020), and given the scale of today's pre-training datasets and LLMs, one would hope that they can learn a huge amount of information from web-sourced text. However, not all of the knowledge on the Internet appears equally often—there is a long-tail of information that appears rarely or only once.

In this work, we explore the relationship between the knowledge learned by an LLM and the information in its pre-training dataset. Specifically, we study how an LLM's ability to answer a question relates to how many documents associated with that question were seen during pre-training. We focus on factoid QA datasets (Joshi et al., 2017; Kwiatkowski et al., 2019), which lets us ground question-answer pairs into concrete subject-object co-occurrences. As an example, for the QA pair (*In what city was the poet Dante born?*, *Florence*), we consider documents where the entities Dante and Florence co-occur as highly relevant. To identify these entity co-occurrences we apply a highly-parallelized entity linking pipeline to trillions of tokens from datasets such as C4 (Raffel et al., 2020), The Pile (Gao et al., 2020), ROOTS (Laurençon et al., 2022), OpenWebText (Gokaslan & Cohen, 2019), and Wikipedia.

We observe a strong correlation between an LM's ability to answer a question and the number of pre-training documents

[1]UNC Chapel Hill [2]Google Research [3]UC Berkeley. Correspondence to: Nikhil Kandpal <nkandpa2@cs.unc.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

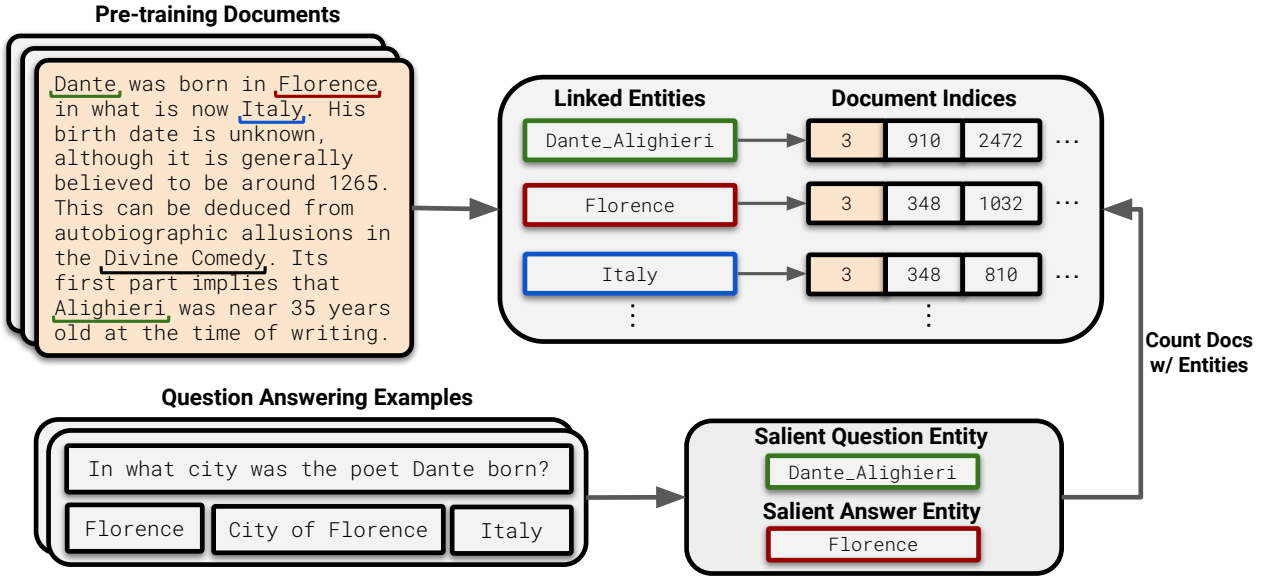


Figure 2. In our document counting pipeline, we first run entity linking on large pre-training datasets (*top left*) and store the set of the document indices in which each entity appears (*top right*). We then entity link downstream QA pairs and extract the salient question and answer entities (*bottom*). Finally, for each question we count the number of documents in which the question and answer entities co-occur.

relevant to that question for numerous QA datasets, pre-training datasets, and model sizes (e.g., Figure 1). For example, the accuracy of BLOOM-176B (Scao et al., 2022) jumps from 25% to above 55% when the number of relevant pre-training documents increases from 10^1 to 10^4.

We also conduct a counterfactual re-training experiment, where we train a 4.8B-parameter LM with and without certain documents. Model accuracy drops significantly on questions whose relevant documents were removed, which validates our entity linking pipeline and shows that the observed correlational trends are likely causal in nature.

Finally, we analyze ways to better capture knowledge that rarely appears in the pre-training data: model scaling and retrieval-augmentation. For model scaling, we find a strong log-linear relationship between parameter count and QA accuracy. These trends show that while scaling up LMs improves knowledge learning, models would need to be scaled dramatically (e.g., to one quadrillion parameters) to achieve competitive QA accuracy on long-tail questions. Retrieval-augmented systems are more promising—when a retriever succeeds in finding a relevant document, it reduces an LLM's need to have a large amount of relevant pre-training text. Nevertheless, retrieval systems themselves still exhibit a mild dependence on relevant document count.

Overall, our work is one of the first to study how LLM knowledge is influenced by pre-training data. To enable future research, we release our code as well as the entity data for ROOTS, The Pile, C4, OpenWebText, and Wikipedia at https://github.com/nkandpa2/long_tail_knowledge.

2. Identifying Relevant Pre-training Data

Background and Research Question Numerous NLP tasks are *knowledge-intensive*: they require recalling and synthesizing facts from a knowledge source (e.g. Wikipedia or the web). Results on knowledge-intensive tasks have been dramatically improved using LLMs, as these models have been shown to leverage the vast amounts of knowledge they learn from their pre-training corpora (Roberts et al., 2020; Petroni et al., 2019; De Cao et al., 2021). However, it remains unclear as to *what kind* of knowledge LMs actually capture—for example, do they simply learn "easy" facts that frequently appear in their pre-training data?

We study this question using closed-book QA evaluations (Roberts et al., 2020) of LLMs in the few-shot setting (Brown et al., 2020). Models are prompted with in-context training examples (QA pairs) and a test question without any relevant background text. The goal of our work is to investigate the relationship between an LM's ability to answer a question and the number of times information relevant to that question appears in the pre-training data.

Our Approach The key challenge is to efficiently identify all of the documents that are relevant to a particular QA pair in pre-training datasets that are hundreds of gigabytes in size. To tackle this, we begin by identifying the salient entities that are contained in a question and its set of ground-truth answer aliases. We then identify relevant pre-training documents by searching for instances where the salient question entity and the answer entity co-occur.

For example, consider the question *In what city was the poet Dante born?* with the valid answers *Florence*, *City of Florence*, and *Italy* (e.g., Figure 2). We extract the salient question and answer entities, Dante_Alighieri and Florence, and count the documents that contain both entities.

Our approach is motivated by Elsahar et al. (2018), who show that when only the subject and object of a subject-object-relation triple co-occur in text, the resulting triple is often also present. In addition, we conduct human studies that show our document counting pipeline selects relevant documents a majority of the time (Section 2.3). Moreover, we further validate our pipeline by training an LM without certain relevant documents and showing that this reduces accuracy on the associated questions (Section 3.2). Based on these findings, we refer to documents that contain the salient question and answer entities as **relevant documents**.

To apply the above method, we must entity link massive pre-training corpora, as well as downstream QA datasets. We accomplish this by building a parallelized pipeline for entity linking (Section 2.1), which we then customize for downstream QA datasets (Section 2.2).

2.1. Entity Linking Pre-training Data

We perform entity linking at scale using a massively distributed run of the DBpedia Spotlight Entity Linker (Mendes et al., 2011), which uses traditional entity linking methods to link entities to DBpedia or Wikidata IDs. We entity link the following pre-training datasets, which were chosen based on their use in the LLMs we consider:

- **The Pile:** an 825GB dataset that contains a mix of 22 different primarily English-language sources (Gao et al., 2020).
- **ROOTS (En):** the 490GB English subset of the ROOTS corpus (Laurençon et al., 2022). Note that we do not study if models trained on the non-English subsets of ROOTS are able to leverage cross-lingual factual knowledge.
- **C4:** a 305GB English corpus that was collected by filtering CommonCrawl (Raffel et al., 2020).
- **OpenWebText:** a 39GB English corpus that contains the text of web pages that were linked on the website Reddit (Gokaslan & Cohen, 2019).
- **Wikipedia:** a text dump of December 2018 Wikipedia articles from Lee et al. (2019), a standard corpus for evaluating open-domain QA systems (e.g. Karpukhin et al. 2020; Lewis et al. 2020; Guu et al. 2020).

For each document in these pre-training datasets, we record the linked entities in a data structure that enables quickly counting individual entity occurrences and entity co-occurrences. This pipeline took approximately 3 weeks to entity link 2.1TB of data on a 128-CPU-core machine.

2.2. Finding Entity Pairs in QA Data

We next entity link two standard open-domain QA datasets: Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). To expand our sample sizes, we use both the training and validation data, except for a small set of examples used for few-shot learning prompts.

We first run the DBPedia entity linker on each example. Because there can be multiple annotated answers for a single example, we concatenate the question and all valid answers, as this enabled more accurate entity linking. We use the most common entity found in the set of ground truth answers as the salient answer entity. We then iterate over all entities found in the question and select the entity that co-occurs the most with the salient answer entity in the pre-training data. In cases where no entity is found in the question, answer, or both, we discard the example. If the resulting number of relevant documents is zero, we discard the example, as this is likely due to an entity linking error.

2.3. Human Evaluation of Document Counting Pipeline

Here, we conduct a human evaluation of our document identification pipeline. Note that a document can vary in the extent to which it is "relevant" to a particular QA pair. For instance, consider the QA pair (*William Van Allan designed which New York building—the tallest brick building in the world in 1930?*, *Chrysler Building*). The documents that we identify as relevant may (1) contain enough information to correctly answer the question, (2) contain information relevant to the question but *not enough* to correctly answer it, or (3) contain no relevant information. For example, a document that mentions that the Chrysler building was designed by William Van Allan, but not that it was the tallest brick building in 1930, would fall into the second category.

We randomly sample 300 QA pairs from TriviaQA and selected one of their relevant documents at random. We then manually labeled the documents into one of the three categories: 33% of documents contained enough information to answer the question and an additional 27% of contained some relevant information. Thus, our pipeline has $\sim$60% precision at identifying relevant documents for TriviaQA.

Our pipeline is imperfect as (1) the entity linker sometimes mis-identifies entities and (2) not all documents containing the salient question and answer entity are relevant. However, when applied at the scale of large-scale pre-training datasets, this pipeline is efficient and achieves enough precision and recall to observe correlational (Section 3.1) and causal (Section 3.2) relationships to QA performance.

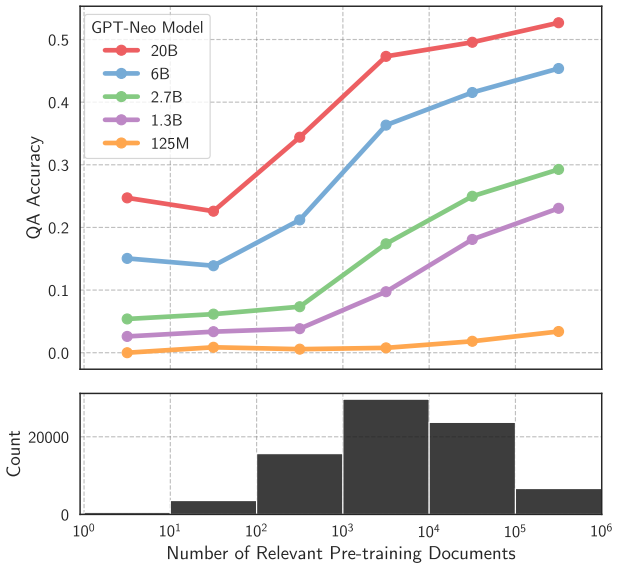


Figure 3. We plot accuracy on TriviaQA versus relevant document count for GPT-Neo. The trends match those seen for BLOOM (Figure 1). We also include a histogram that shows how many QA examples fall into each bucket; TriviaQA often asks about knowledge represented 10^2 to 10^5 times in the pre-training data.

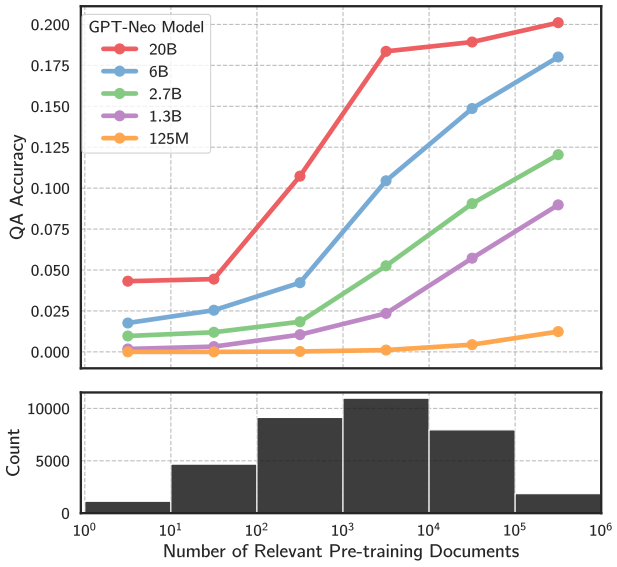


Figure 4. We plot accuracy on Natural Questions versus relevant document count for GPT-Neo. The trends match those in TriviaQA—model accuracy is highly dependent on fact count.

3. LM Accuracy Depends on Relevant Document Count

In this section, we measure the relationship between an LLM's ability to answer a question and the number of relevant documents in the pre-training corpus. We use popular Transformer decoder-only LMs (Vaswani et al., 2017) that span three orders of magnitude in size:

- **GPT-Neo:** The GPT-Neo, GPT-NeoX, and GPT-J LMs trained by EleutherAI on the Pile (Gao et al., 2020) that range in size from 125M to 20B parameters (Black et al., 2021; Wang & Komatsuzaki, 2021; Black et al., 2022). We refer to these models collectively as GPT-Neo models.
- **BLOOM:** Models trained by the BigScience initiative on the ROOTS dataset (Scao et al., 2022). The BLOOM models are multi-lingual; we analyze their English performance only. The models range in size from 560M to 176B parameters.
- **GPT-3:** Models trained by OpenAI that range in size from ≈350M (Ada) to ≈175B parameters (Davinci). Since the pre-training data for these models is not public, we estimate relevant document counts by scaling up the counts from OpenWebText to simulate if the dataset was the same size as GPT-3's pre-training data. We recognize that there is uncertainty around these models' pre-training data, their exact sizes, and whether they have been fine-tuned. We therefore report these results in the Appendix for readers to interpret with these sources of error in mind.

We use these LMs because (with the exception of GPT-3) they are the largest open-source models for which the pre-training data is publicly available. We focus on 4-shot evaluation, although we found that other amounts of in-context training examples produced similar trends. We use simple prompts consisting of templates of the form

Q: [In-Context Question 1]
A: [In-Context Answer 1]
⋮
Q: [In-Context Question n]
A: [In-Context Answer n]
Q: [Test Question]

We generate answers by greedy decoding until the models generate a newline character, and we evaluate answers using the standard Exatch Match (EM) metric against the ground-truth answer set (Rajpurkar et al., 2016).

3.1. Correlational Analysis

We first evaluate the BLOOM and GPT-Neo model families on TriviaQA and plot their QA accuracies versus the number of relevant documents in Figures 1 and 3. For improved readability, we average the accuracy for QA pairs using log-spaced bins (e.g., the accuracy for all questions with 1 to 10 relevant documents, 10 to 100 relevant documents, etc.). Below each plot, we also include a histogram that shows how many QA examples fall into each bin. We trim the plots when the bins contain fewer than 500 QA examples to avoid reporting accuracies for small sample sizes.

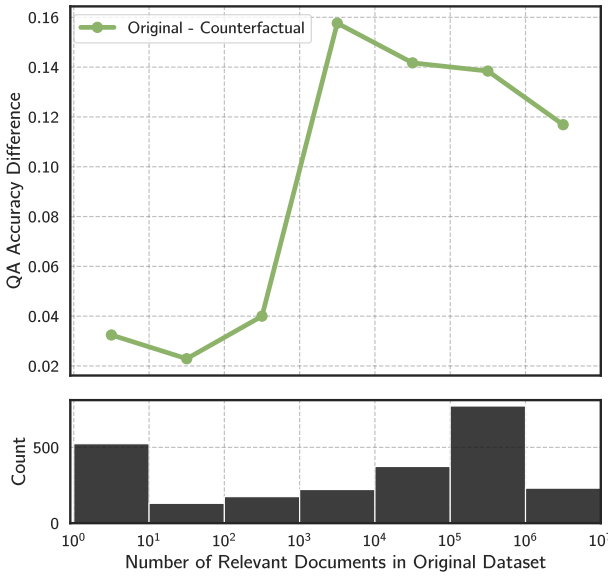


Figure 5. We run a counterfactual experiment, where we re-train an LM without certain documents. We take TriviaQA questions with different document counts and delete all of their relevant pre-training documents. The difference in accuracy between the original model and the re-trained LM (*counterfactual*) is high when the original number of relevant documents is large.

There is a strong correlation between question answering accuracy and relevant document count for all tested models. Correspondingly, when the number of relevant documents is low, models are quite inaccurate, e.g., the accuracy of BLOOM-176B jumps from 25% to above 55% when the number relevant documents increases from 10^1 to 10^4. Model size is also a major factor in knowledge learning: as the number of model parameters is increased, the QA performance substantially improves. For example, BLOOM-176B has over $4\times$ higher accuracy than BLOOM-560M on TriviaQA questions with more than 10^5 relevant documents.

We repeat this experiment using the Natural Questions QA dataset and find similar trends for all model families (see Figure 4 for GPT-Neo, and Figures 10 and 11 in the Appendix for BLOOM and GPT-3 results).

Simpler Methods for Identifying Relevant Documents Are Less Effective In the experiments above, we identify relevant documents by searching for co-occurrences of salient question and answer entities. To evaluate whether this process is necessary, we compare against two baseline document identification methods: counting documents that contain the salient question entity and counting documents that contain the salient answer entity (as done in Petroni et al. 2019).

We show in Figure 13 that all three document identification methods are correlated with QA accuracy. However, when only considering QA examples where the question and answer entities co-occur few (< 5) times, the two baseline methods no longer correlate with QA accuracy. This indicates that counting documents with just the answer entity or question entity alone is insufficient for explaining why LMs are able to answer certain questions. This validates our definition of relevant documents as those that contain both the question entity and answer entity.

Humans Show Different Trends Than LMs An alternate explanation for our results is that questions with lower document counts are simply "harder", which causes the drop in model performance. We show that this is not the case by measuring human accuracy on Natural Questions. We use a leave-one-annotator-out metric, where we take questions that are labeled by 5 different human raters (all of whom can see the necessary background text), hold out one of the raters, and use the other four as the ground-truth answer set. We plot the human accuracy versus relevant document count in the top of Figure 7. Human accuracy is actually highest for the questions with *few* relevant documents, the opposite trend of models. We hypothesize that humans are better on questions with few relevant documents because (1) questions about rarer facts are more likely to be simple factoids compared to common entities, and (2) the Wikipedia documents are that are provided to the annotators are shorter for rarer entities, which makes reading comprehension easier and increases inner-annotator agreement.

3.2. Causal Analysis via Re-training

Our results thus far are correlational in nature: there may be unknown confounds that explain them away, i.e., the rarer questions are more difficult for LMs for other reasons. Here we establish a causal relationship by removing certain documents in the training data and re-training the LM.

We first train a baseline 4.8 billion parameter LM on C4, following the setup from Wang et al. (2022). We then measure the effect of deleting certain documents from the training set. For each log-scaled bin of relevant document count (e.g., 10^0 to 10^1 relevant documents, 10^1 to 10^2, ...) we sample 100 questions from Trivia QA and remove all relevant documents for those questions in C4. In total, this removes about 30% of C4. Finally, we train a "counterfactual" LM on this modified pre-training dataset and compare its performance to the baseline model. For both the baseline model and the counterfactual model, we train for a single epoch. Note that the counterfactual model was trained for 30% fewer steps, which makes it slightly worse in performance overall. To account for this, we only study the performance on questions whose relevant documents were removed.

We show the difference in performance between the two LMs on questions whose documents were removed in Fig-

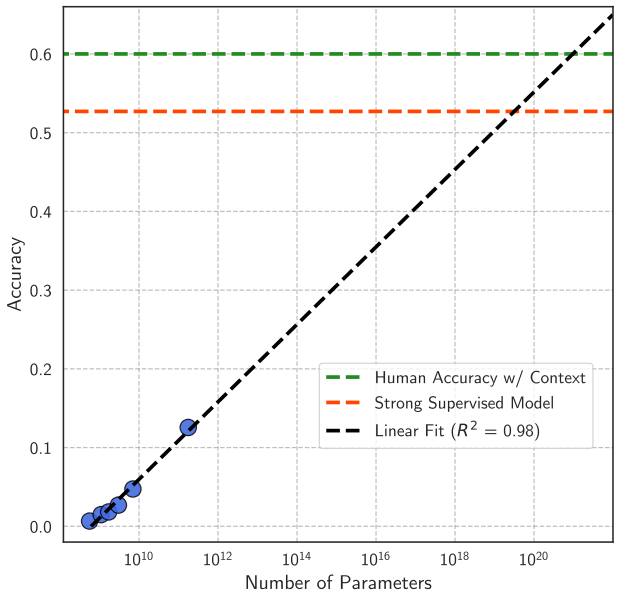


Figure 6. Scaling trends for fact learning. We plot BLOOM accuracy on rare instances from Natural Questions (< 100 relevant docs) as a function of the log of the model size. Extrapolating from the empirical line of best fit—which approximates the trend well at $R^2 = 0.98$—implies that immensely large models would be necessary to get high accuracy.

ure 5. For questions with few relevant documents in the original C4 dataset, performance is poor for both the baseline and the counterfactual LM, i.e., their performance difference is small. However, for questions with many relevant documents, performance is significantly worse for the counterfactual LM. This suggests a *causal* link between the number of relevant documents and QA performance.

4. Methods to Improve Rare Fact Learning

Thus far, we showed that LLMs have a strong dependence on relevant document count. Here, we investigate methods to mitigate this dependence: increasing data scale, increasing model scale, and adding an auxiliary retrieval module.

4.1. Can We Scale Up Datasets?

Today's largest LLMs are pre-trained on hundreds of billions of tokens. One naïve approach for improving accuracy on questions about less-prevalent knowledge is to collect larger quantities of data. Our results suggest that this would not significantly improve accuracy as scaling datasets by moderate factors (e.g., 5x) usually results in small accuracy gains. An alternative idea would be to increase the diversity of the pre-training data. However, we also believe this would provide minimal benefit because many data sources are surprisingly correlated . Although each of the pre-training datasets considered were collected independently, the amount of supporting information they provide for different TriviaQA examples is highly consistent as seen by the rank correlations between their relevant document counts in Table 1.

	ROOTS	Pile	C4	OWT	Wiki
ROOTS	-	0.97	0.97	0.94	0.87
Pile	-	-	0.95	0.96	0.87
C4	-	-	-	0.96	0.90
OWT	-	-	-	-	0.91
Wiki	-	-	-	-	-

Table 1. Spearman rank correlations of the relevant document counts for TriviaQA examples in The Pile, ROOTS, C4, OpenWebText, and Wikipedia. Despite having different collection methodologies, these pre-training datasets are highly correlated in terms of how much information they contain related to different QA pairs.

4.2. Can We Scale Up Models?

Using larger models consistently produces better QA performance. However, our results suggest that one would need immensely large LMs to achieve high accuracy on long-tail questions. In Figure 6 we plot a scaling trend line for rare fact learning, where we show BLOOM accuracy on rare instances from Natural Questions (< 100 relevant docs) as a function of the log of the model size. The empirical log-linear trend—which approximates the scaling extremely well ($R^2 = 0.98$)—shows that in order to match a strong supervised baseline (Izacard & Grave, 2021) or human performance, one would need a BLOOM model with over 10^{18} (one quintillion) parameters.[1] We see similar trends for other models and datasets (see Figure 12 in the Appendix).

Modifying the Training Objective Another option similar to scaling up models is to directly modify the training objective to encourage memorization. One simple method to accomplish this is to increase the number of training epochs. All of the LMs that we study do limited epochs, as it is generally seen as preferable to use large enough pre-training datasets so that the LM completes one epoch of training when the compute budget is exhausted (Raffel et al., 2020). However, in the context of QA, it may be preferable to increase epochs and reduce data size to ensure models memorize as much as possible. Alternatively, one could consider modifying the training loss to encourage the model to focus on salient facts (Guu et al., 2020) or designing a curriculum to minimize forgetting (Jagielski et al., 2023).

[1]For this experiment, the supervised and human accuracies are computed over the validation set whereas the scaling trend is computed using the train and validation sets.

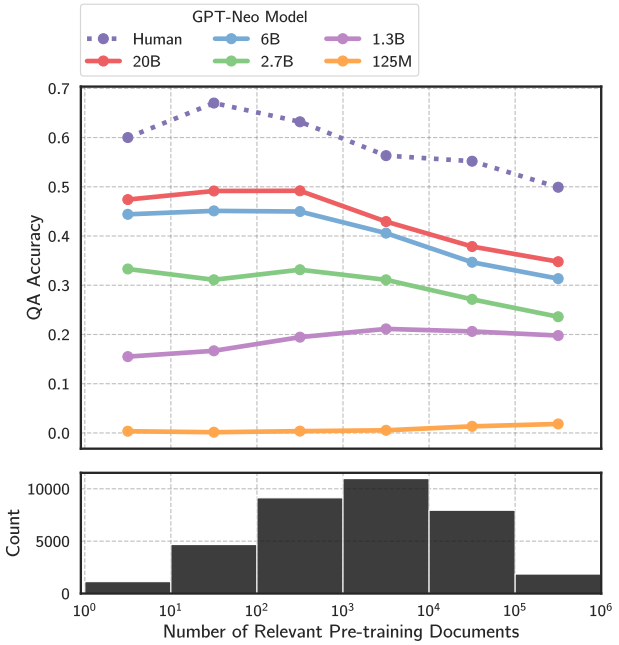


Figure 7. Models with access to the required background context do not struggle on questions with low relevant document count. Concretely, we provide questions and gold paragraphs to GPT-Neo models on Natural Questions, and their accuracy trends roughly match the trends of humans.

4.3. Can We Use Retrieval Augmentation?

Thus far, we use LMs as isolated systems that do not leverage external information. However, for knowledge-intensive tasks, a natural alternative is to make LMs *retrieval-augmented*, i.e., combine them with a retrieval module that returns relevant textual contexts (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020). Here, we study whether retrieval-augmented models can mitigate the dependence on the amount of relevant knowledge in the pre-training data.

Oracle Retrieval We first study an oracle setting where we provide LMs with a gold paragraph from Wikipedia that supports each answer in Natural Questions (Petroni et al., 2020). We use the 300-word segment that surrounds the ground-truth answer from the gold Wikipedia page and evaluate the 2-shot accuracy of GPT-Neo. Figure 7 shows that oracle retrieval-augmentation dramatically boosts accuracy over closed-book models, especially on rarer instances. Similar to Liu et al. (2022), we also find that QA accuracy actually goes *down* as the number of relevant documents increases—the opposite trend of closed-book LLMs. As discussed in Section 3.1, humans exhibit the same trend, likely because rare questions are easier on average when relevant context information.

BM25 Retrieval We next follow a common retrieval-augmented baseline, where we use a BM25 retriever (Robertson & Zaragoza, 2009) to select paragraphs from Wikipedia. We add the top-3 highest scoring paragraphs into the prompt for both the in-context training examples and the test question. We verify that at least one of the retrieved paragraphs contains the answer for each in-context training example, to ensure that the LM learns to utilize on the documents.

We first evaluate the BM25 retriever's top-k recall on its knowledge corpus (Wikipedia) as a function of relevant document count, and plot the results in Figure 8. We find that BM25 attains reasonably high recall, especially for larger values of k. However, the BM25 retriever still shows a mild dependence on relevant document count. We next evaluate the accuracy of BM25-augmented GPT-Neo models on Natural Questions and plot the results in Figure 9. Overall, retrieval-augmented models outperform their closed-book counterparts across all ranges of relevant document counts, and especially on rare examples. These results suggest that retrieval augmentation provides a promising path towards improving performance on questions with few relevant documents in the pre-training dataset.

5. Related Work

Identifying The Origins of Few-shot Learning Our work contributes to an emerging line of research that explains the success of zero- and few-shot learning in language models by tracing their behavior back to the pre-training data. For example, Razeghi et al. (2022) show mathematical reasoning capabilities can be correlated with training data frequency, and Shin et al. (2022) and Han & Tsvetkov (2022) show that training corpus source can influence few-shot accuracies.

The most similar work to ours in this context is Elazar et al. (2022), who use causal inference to measure the effect of pre-training data statistics on QA performance. Their main focus is testing the extent to which LMs answer questions using heuristics based on co-occurrences between subjects, objects, and textual patterns in the pre-training data. Our main focus is to measure the relationship between the knowledge learned by an LLM and the prevalence of that knowledge in the pre-training data. Moreover, we also conduct re-training experiments and study how model scaling and retrieval-augmentation affect knowledge learning.

Memorization and Privacy Past work studies training data memorization from the perspective of privacy, i.e., how LMs inadvertently reveal private text (Carlini et al., 2019; 2021; Lee et al., 2021). These works focus on how LMs memorize and repeat *verbatim* text samples, and the effect of duplicating those texts in the training set (Kandpal et al.,

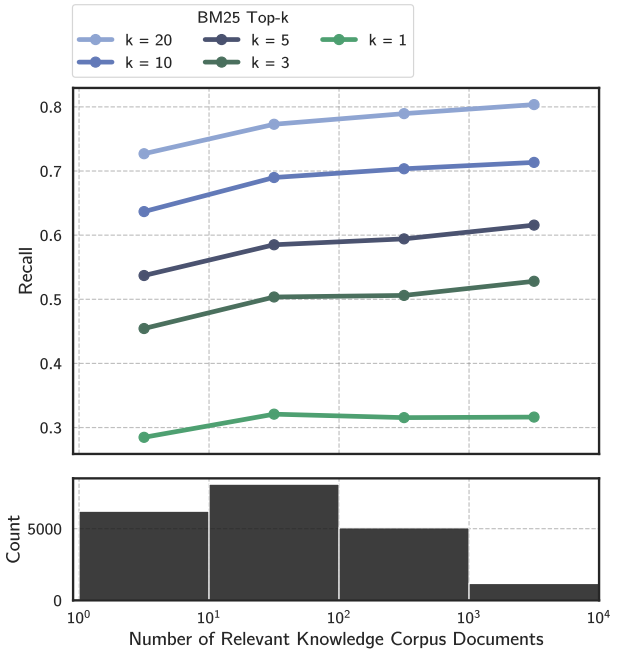


Figure 8. Retrieval systems such as BM25 have a mild dependence on document count. Above we plot the top-k recall for BM25 on Natural Questions for different values of k.

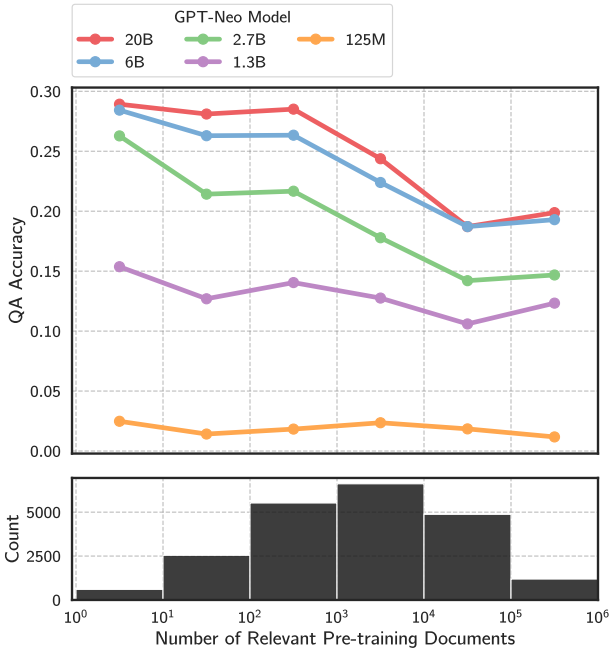


Figure 9. Retrieval-augmented LMs no longer exhibit low accuracy on rare instances. We plot GPT-Neo accuracy on Natural Questions when augmented with three paragraphs from BM25.

2022). Doing so has various limitations, as memorization can be harmful or beneficial even in non-verbatim cases (Ippolito et al., 2022). Our work takes studies non-verbatim memorization in the context of QA—our LMs memorize facts in text form and then answers questions about those facts at test time.

Memorization and Fact Learning Existing work also analyzes the relationship between the pre-training data and the factual knowledge of LLMs. Akyürek et al. (2022) look to automatically identify which documents were most influential for a language model's QA predictions. Our work instead directly identifies and estimates the number of relevant documents via entity linking large corpora. Other work notices a correspondence between model accuracy and data frequency for different knowledge-intensive tasks (Petroni et al., 2019; Kassner et al., 2020; De Cao et al., 2021; Wei et al., 2021; Févry et al., 2020) and for domains outside of NLP (Rao et al., 2021). Our paper reports similar findings, but scales this analysis to massive LM pre-training datasets and model sizes.

In concurrent and independent work, Mallen et al. (2022) study how QA performance correlates with frequency in the pre-training data. Unlike our work, they do not use entity linking methods to count occurrences and instead use proxies such as entity popularity on Wikipedia. They also find QA accuracy is highly correlated with pre-training data frequency and show that retrieval models can improve long-tail knowledge. Our work differs in that we conduct causal re-training experiments and find that model scaling is highly beneficial to long-tail QA performance.

6. Conclusion and Future Work

Large language models demonstrate impressive few-shot learning capabilities that arise from simply training on large-scale internet text. With the open-source release of LLMs—and their associated pre-training datasets—the research community can now begin to understand the origins of these capabilities. Our work is one of the first to relate an observed phenomenon in LLMs back to the pre-training data itself. In our case, our results are negative: while LLMs achieve moderate performance on open-domain QA benchmarks, they are mainly successful on questions that probe knowledge that appears widely in their pre-training datasets.

Our work raises numerous directions for further inquiry, namely, how to improve retention of long-tail knowledge given that simply scaling up model and dataset size will likely be insufficient. We are personally excited about improving retrieval-augmented LMs, especially with regards to their efficiency and retrieval accuracy. Moreover, our work focuses on knowledge learning as it relates to factoid question answering, but we leave open the question as to whether similar relationships exist for other types of

tasks, be it knowledge-intensive or otherwise. Relatedly, even though our work analyzes the impact of memorization on question answering, our results may have implications for other tasks that require using (or avoiding) memorized knowledge, e.g., analyzing private text, performing commonsense reasoning, or predicting source code. Finally, we hope that future evaluations of few-shot learning can continue to shed light into model behavior by tracing accuracy back to properties of the pre-training data. In particular, our work shows that by performing such an analysis, one can help elucidate the successes and failures of existing models, as well as help to identify possible paths forward to improve today's systems.

Acknowledgements

We thank Sewon Min, Sameer Singh, Katherine Lee, and the members of UNC NLP for their valuable feedback. Eric Wallace is supported by the Apple Scholars in AI/ML Fellowship. This work was supported by NSF-AI Engage Institute DRL-2112635.

References

Akyürek, E., Bolukbasi, T., Liu, F., Xiong, B., Tenney, I., Andreas, J., and Guu, K. Tracing knowledge in language models back to the training data. In *Findings of EMNLP*, 2022.

Black, S., Leo, G., Wang, P., Leahy, C., and Biderman, S. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. GPT-Neox-20B: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, 2019.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.

De Cao, N., Izacard, G., Riedel, S., and Petroni, F. Autoregressive entity retrieval. In *ICLR*, 2021.

Elazar, Y., Kassner, N., Ravfogel, S., Feder, A., Ravichander, A., Mosbach, M., Belinkov, Y., Schütze, H., and Goldberg, Y. Measuring causal effects of data statistics on language model's factual predictions. *arXiv preprint arXiv:2207.14251*, 2022.

Elsahar, H., Vougiouklis, P., Remaci, A., Gravier, C., Hare, J., Laforest, F., and Simperl, E. T-REx: A large scale alignment of natural language with knowledge base triples. In *LREC*, 2018.

Févry, T., Soares, L. B., FitzGerald, N., Choi, E., and Kwiatkowski, T. Entities as experts: Sparse memory access with entity supervision. In *EMNLP*, 2020.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gokaslan, A. and Cohen, V. Openwebtext corpus, 2019.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *ICML*, 2020.

Han, X. and Tsvetkov, Y. ORCA: Interpreting prompted language models via locating supporting data evidence in the ocean of pretraining data. *arXiv preprint arXiv:2205.12600*, 2022.

Ippolito, D., Tramèr, F., Nasr, M., Zhang, C., Jagielski, M., Lee, K., Choquette-Choo, C. A., and Carlini, N. Preventing verbatim memorization in language models gives a false sense of privacy. *arXiv preprint arXiv:2210.17546*, 2022.

Izacard, G. and Grave, E. Distilling knowledge from reader to retriever for question answering. In *ICLR*, 2021.

Jagielski, M., Thakkar, O., Tramer, F., Ippolito, D., Lee, K., Carlini, N., Wallace, E., Song, S., Thakurta, A., Papernot, N., et al. Measuring forgetting of memorized training examples. In *ICLR*, 2023.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, 2017.

Kandpal, N., Wallace, E., and Raffel, C. Deduplicating training data mitigates privacy risks in language models. In *ICML*, 2022.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

Kassner, N., Krojer, B., and Schütze, H. Are pretrained language models symbolic reasoners over knowledge? In *CoNLL*, 2020.

Kwiatkowski, T., Palomaki, J., Rhinehart, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., et al. Natural Questions: A benchmark for question answering research. In *TACL*, 2019.

Laurençon, H., Saulnier, L., Wang, T., Akiki, C., del Moral, A. V., Scao, T. L., Werra, L. V., Mou, C., Ponferrada, E. G., Nguyen, H., Frohberg, J., Šaško, M., Lhoest, Q., McMillan-Major, A., et al. The BigScience ROOTS corpus: A 1.6TB composite multilingual dataset. In *NeurIPS*, 2022.

Lee, K., Chang, M.-W., and Toutanova, K. Latent retrieval for weakly supervised open domain question answering. In *ACL*, 2019.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. In *ACL*, 2021.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020.

Liu, L., Lewis, P., Riedel, S., and Stenetorp, P. Challenges in generalization in open domain question answering. In *Findings of NAACL*, 2022.

Mallen, A., Asai, A., Zhong, V., Das, R., Hajishirzi, H., and Khashabi, D. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 2022.

Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. DBpedia Spotlight: Shedding light on the web of documents. In *International Conference on Semantic Systems*, 2011.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. Language models as knowledge bases? In *EMNLP*, 2019.

Petroni, F., Lewis, P. S. H., Piktus, A., Rocktäschel, T., Wu, Y., Miller, A. H., and Riedel, S. How context affects language models' factual predictions. In *AKBC*, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. In *JMLR*, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Rao, R. M., Liu, J., Verkuil, R., Meier, J., Canny, J., Abbeel, P., Sercu, T., and Rives, A. Msa transformer. In *ICML*, 2021.

Razeghi, Y., Logan IV, R. L., Gardner, M., and Singh, S. Impact of pretraining term frequencies on few-shot reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model? In *EMNLP*, 2020.

Robertson, S. and Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in IR*, 2009.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E.-J., Ili'c, S., Hesslow, D., Castagn'e, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., et al. BLOOM: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Shin, S., Lee, S.-W., Ahn, H., Kim, S., Kim, H., Kim, B., Cho, K., Lee, G., Park, W., Ha, J.-W., et al. On the effect of pretraining corpora on in-context learning by a large-scale language model. In *NAACL*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, 2021.

Wang, T., Roberts, A., Hesslow, D., Scao, T. L., Chung, H. W., Beltagy, I., Launay, J., and Raffel, C. What language model architecture and pretraining objective work best for zero-shot generalization? In *ICML*, 2022.

Wei, J., Garrette, D., Linzen, T., and Pavlick, E. Frequency effects on syntactic rule learning in transformers. In *EMNLP*, 2021.

A. Additional Results: Relevant Document Scaling

Here we show how QA performance is related to the number of relevant pre-training documents for the BLOOM on Natural Questions (Figure 10) and the GPT-3 model family on TriviaQA and Natural Questions (Figure 11). Like the results in the main text, models are significantly better at answering questions about facts that are well supported in the pre-training data and model scale improves knowledge acquisition.

Note that our estimates for the number of relevant pre-training documents for the GPT-3 model family may be inaccurate since the training data for GPT-3 is not public. Instead, we estimate these relevant document counts using the open source dataset OpenWebText, which was collected with a similar process to the reported collection methodology for the GPT-3 pre-training dataset.

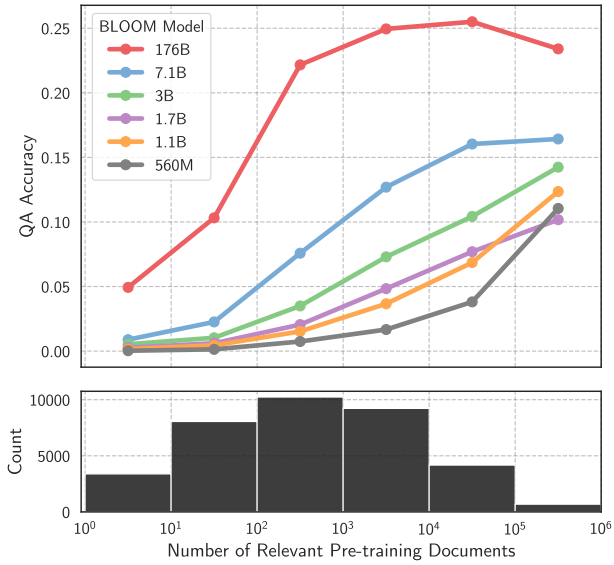
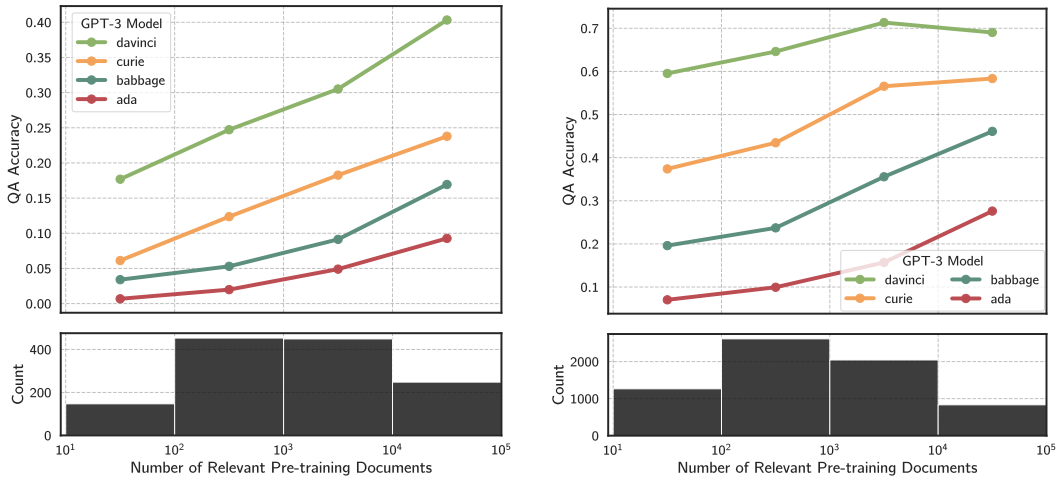


Figure 10. We show results for Natural Questions for BLOOM. The trends match those seen in TriviaQA, although the accuracy is lower overall for Natural Questions.

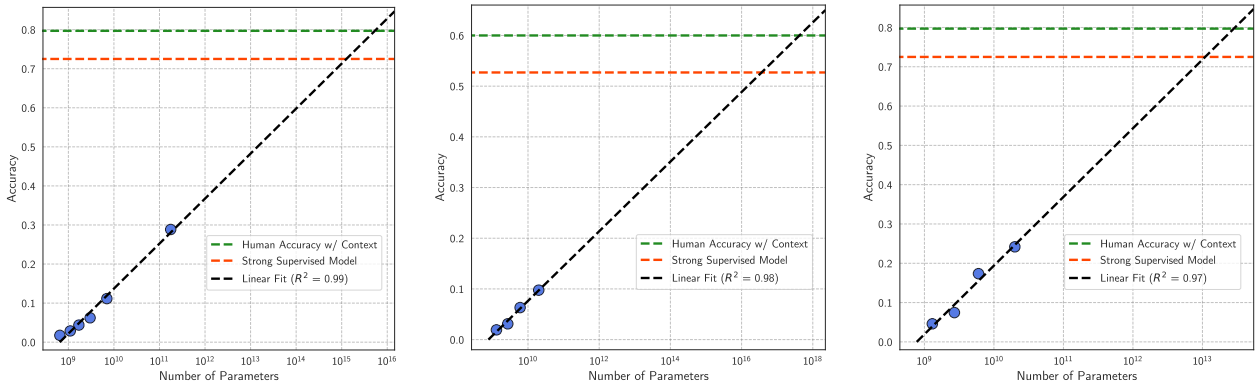


(a) (b)

Figure 11. We present the QA results for GPT-3, with Natural Questions shown in **(a)** and TriviaQA shown in **(b)**. The trends match those seen in BLOOM and GPT-Neo. Note that our estimates for the number of relevant pre-training documents may be inaccurate because the training data for GPT-3 is not public.

B. Additional Results: Model Scaling

In this section we show additional results for how long-tail QA accuracy scales with model size for the BLOOM model family on TriviaQA and the GPT-Neo model family on Natural Questions and TriviaQA (Figure 12). The log-linear trend matches the results shown in the main text.

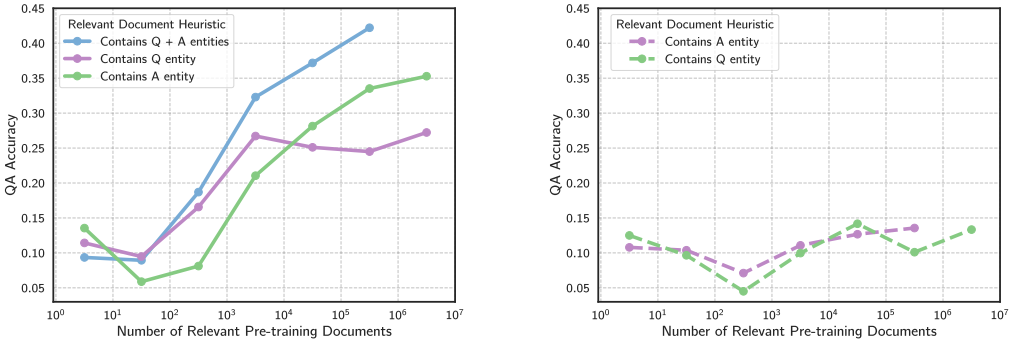


(a) (b) (c)

Figure 12. We present additional scaling laws for BLOOM on TriviaQA **(a)**, and GPT-Neo on Natural Questions **(b)** and TriviaQA **(c)**. All the trends are similar—we will need to scale up models dramatically to reach high QA accuracy—but the exact degree to how much we would need to scale models changes across the different settings.

C. Relevant Document Counting Heuristics

In this section, we analyze the difference between our relevant document heuristic, which counts documents where the salient question and answer entity co-occur, compared to two simple baselines: counting documents containing the question entity and documents containing the answer entity. In Figure 13(a) we show that all three document counting heuristics are correlated with QA accuracy. However, as seen in Figure 13(b) the correlation of the two baseline counting methods with QA accuracy disappears when only considering QA examples where the question and answer entity co-occur few (< 5) times in the pre-training data. Thus, these baseline counting heuristics appear correlated with QA accuracy simply because they are simply correlated with question and answer entity co-occurrence (i.e., common entities tend to co-occur with other entities more frequently) rather than causally related to QA performance.



(a) (b)

Figure 13. In **(a)**, we plot the relationship between model accuracy and the count of the question entity alone, as well as the answer entity alone. QA accuracy increases as both of these counts increase. In **(b)**, we consider only f QA pairs with few question and answer entity co-occurrences (< 5 documents). For this subpopulation of QA pairs, neither of the baseline heuristics are correlated with QA accuracy.

Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

Pengfei Liu
Carnegie Mellon University
pliu3@cs.cmu.edu

Weizhe Yuan
Carnegie Mellon University
weizhey@cs.cmu.edu

Jinlan Fu
National University of Singapore
jinlanjonna@gmail.com

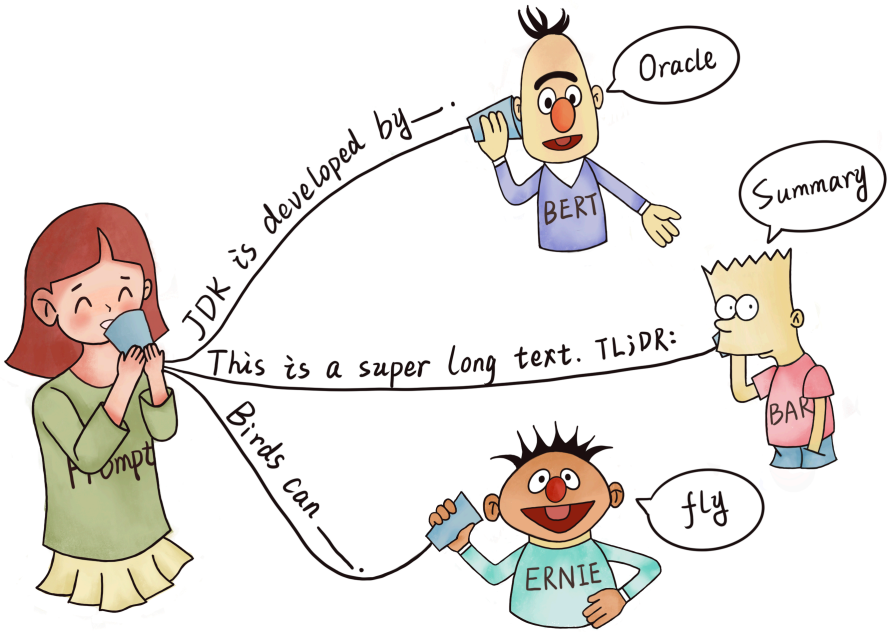
Zhengbao Jiang
Carnegie Mellon University
zhengbaj@cs.cmu.edu

Hiroaki Hayashi
Carnegie Mellon University
hiroakih@cs.cmu.edu

Graham Neubig
Carnegie Mellon University
gneubig@cs.cmu.edu

Abstract

This paper surveys and organizes research works in a new paradigm in natural language processing, which we dub "prompt-based learning". Unlike traditional supervised learning, which trains a model to take in an input $\boldsymbol{x}$ and predict an output $\boldsymbol{y}$ as $P(\boldsymbol{y}|\boldsymbol{x})$, prompt-based learning is based on language models that model the probability of text directly. To use these models to perform prediction tasks, the original input $\boldsymbol{x}$ is modified using a *template* into a textual string *prompt* $\boldsymbol{x}'$ that has some unfilled slots, and then the language model is used to probabilistically fill the unfilled information to obtain a final string $\hat{\boldsymbol{x}}$, from which the final output $\boldsymbol{y}$ can be derived. This framework is powerful and attractive for a number of reasons: it allows the language model to be *pre-trained* on massive amounts of raw text, and by defining a new prompting function the model is able to perform *few-shot* or even *zero-shot* learning, adapting to new scenarios with few or no labeled data. In this paper we introduce the basics of this promising paradigm, describe a unified set of mathematical notations that can cover a wide variety of existing work, and organize existing work along several dimensions, e.g. the choice of pre-trained models, prompts, and tuning strategies. To make the field more accessible to interested beginners, we not only make a systematic review of existing works and a highly structured typology of prompt-based concepts, but also release other resources, e.g., a website NLPedia–Pretrain including constantly-updated survey, and paperlist.



Contents

1 Two Sea Changes in NLP

Fully supervised learning, where a task-specific model is trained solely on a dataset of input-output examples for the target task, has long played a central role in many machine learning tasks (Kotsiantis et al., 2007), and natural language processing (NLP) was no exception. Because such fully supervised datasets are ever-insufficient for learning high-quality models, early NLP models relied heavily on *feature engineering* (Tab. 1 a.; e.g. Lafferty et al. (2001); Guyon et al. (2002); Och et al. (2004); Zhang and Nivre (2011)), where NLP researchers or engineers used their domain knowledge to define and extract salient features from raw data and provide models with the appropriate inductive bias to learn from this limited data. With the advent of neural network models for NLP, salient features were learned jointly with the training of the model itself (Collobert et al., 2011; Bengio et al., 2013), and hence focus shifted to *architecture engineering*, where inductive bias was rather provided through the design of a suitable network architecture conducive to learning such features (Tab. 1 b.; e.g. Hochreiter and Schmidhuber (1997); Kalchbrenner et al. (2014); Chung et al. (2014); Kim (2014); Bahdanau et al. (2014); Vaswani et al. (2017)).[1]

However, from 2017-2019 there was a sea change in the learning of NLP models, and this fully supervised paradigm is now playing an ever-shrinking role. Specifically, the standard shifted to the *pre-train and fine-tune* paradigm (Tab. 1 c.; e.g. Radford and Narasimhan (2018); Peters et al. (2018); Dong et al. (2019); Yang et al. (2019); Lewis et al. (2020a)). In this paradigm, a model with a fixed[2] architecture is *pre-trained* as a language model (LM), predicting the probability of observed textual data. Because the raw textual data necessary to train LMs is available in abundance, these LMs can be trained on large datasets, in the process learning robust general-purpose features of the language it is modeling. The above pre-trained LM will be then adapted to different downstream tasks by introducing additional parameters and *fine-tuning* them using task-specific objective functions. Within this paradigm, the focus turned mainly to *objective engineering*, designing the training objectives used at both the pre-training and fine-tuning stages. For example, Zhang et al. (2020a) show that introducing a loss function of predicting salient sentences from a document will lead to a better pre-trained model for text summarization. Notably, the main body of the pre-trained LM is generally (but not always; Peters et al. (2019)) fine-tuned as well to make it more suitable for solving the downstream task.

Now, as of this writing in 2021, we are in the middle of a second sea change, in which the "pre-train, fine-tune" procedure is replaced by one in which we dub "*pre-train, prompt, and predict*". In this paradigm, instead of adapting pre-trained LMs to downstream tasks via objective engineering, downstream tasks are reformulated to look more like those solved during the original LM training with the help of a textual *prompt*. For example, when recognizing the emotion of a social media post, "I missed the bus today.", we may continue with a prompt "I felt so ___", and ask the LM to fill the blank with an emotion-bearing word. Or if we choose the prompt "English: I missed the bus today. French: ___"), an LM may be able to fill in the blank with a French translation. In this way, by selecting the appropriate prompts we can manipulate the model behavior so that the pre-trained LM itself can be used to *predict* the desired output, sometimes even without any additional task-specific training (Tab. 1 d.; e.g. Radford et al. (2019); Petroni et al. (2019); Brown et al. (2020); Raffel et al. (2020); Schick and Schütze (2021b); Gao et al. (2021)). The advantage of this method is that, given a suite of appropriate prompts, a single LM trained in an entirely unsupervised fashion can be used to solve a great number of tasks (Brown et al., 2020; Sun et al., 2021). However, as with most conceptually enticing prospects, there is a catch – this method introduces the necessity for *prompt engineering*, finding the most appropriate prompt to allow a LM to solve the task at hand.

This survey attempts to organize the current state of knowledge in this rapidly developing field by providing an overview and formal definition of prompting methods (§2), and an overview of the pre-trained language models that use these prompts (§3). This is followed by in-depth discussion of prompting methods, from basics such as prompt engineering (§4) and answer engineering (§5) to more advanced concepts such as multi-prompt learning methods (§6) and prompt-aware training methods (§7). We then organize the various applications to which prompt-based learning methods have been applied, and discuss how they interact with the choice of prompting method (§8). Finally, we attempt to situate the current state of prompting methods in the research ecosystem, making connections to other research fields (§9), suggesting some current challenging problems that may be ripe for further research (§10), and performing a meta-analysis of current research trends (§11).

Finally, in order to help beginners who are interested in this field learn more effectively, we highlight some systematic resources about prompt learning (as well as pre-training) provided both within this survey and on companion websites:

-  : A website of prompt-based learning that contains: frequent updates to this survey, related slides, etc.
- Fig.1: A typology of important concepts for prompt-based learning.

[1] Even during this stage, there was some use of pre-trained models exemplified by word2vec (Mikolov et al., 2013b,a) and GloVe (Pennington et al., 2014), but they were used for only a limited portion of the final model parameters.

[2] This paradigm is less conducive to architectural exploration because (i) unsupervised pre-training allows models to learn with fewer structural priors, and (ii) as pre-training of models is time-consuming, experimenting with structural variants is costly.

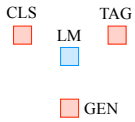
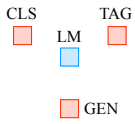
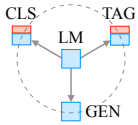
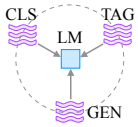
Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Table 1: Four paradigms in NLP. The "**engineering**" column represents the type of engineering to be done to build strong systems. The "**task relation**" column, shows the relationship between language models (LM) and other NLP tasks (CLS: classification, TAG: sequence tagging, GEN: text generation). : fully unsupervised training. : fully supervised training. : Supervised training combined with unsupervised training. indicates a textual prompt. Dashed lines suggest that different tasks can be connected by sharing parameters of pre-trained models. "LM→Task" represents *adapting LMs (objectives) to downstream tasks* while "Task→LM" denotes *adapting downstream tasks (formulations) to LMs*.

- Tab.7: A systematic and comprehensive comparison among different prompting methods.
- Tab.10: An organization of commonly-used prompts.
- Tab.12: A timeline of prompt-based research works.
- Tab.13: A systematic and comprehensive comparison among different pre-trained LMs.

2 A Formal Description of Prompting

2.1 Supervised Learning in NLP

In a traditional supervised learning system for NLP, we take an **input** $\boldsymbol{x}$, usually text, and predict an **output** $\boldsymbol{y}$ based on a model $P(\boldsymbol{y}|\boldsymbol{x};\theta)$. $\boldsymbol{y}$ could be a label, text, or other variety of output. In order to learn the parameters θ of this model, we use a dataset containing pairs of inputs and outputs, and train a model to predict this conditional probability. We will illustrate this with two stereotypical examples.

First, *text classification* takes an input text $\boldsymbol{x}$ and predicts a label y from a fixed label set $\mathcal{Y}$. To give an example, sentiment analysis (Pang et al., 2002; Socher et al., 2013) may take an input $\boldsymbol{x}$ ="I love this movie." and predict a label $y = ++$, out of a label set $\mathcal{Y} = \{++, +, \sim, -, --\}$.

Second, *conditional text generation* takes an input $\boldsymbol{x}$ and generates another text $\boldsymbol{y}$. One example is machine translation (Koehn, 2009), where the input is text in one language such as the Finnish $\boldsymbol{x}$ = "Hyvää huomenta." and the output is the English $\boldsymbol{y}$ = "Good morning"..

2.2 Prompting Basics

The main issue with supervised learning is that in order to train a model $P(\boldsymbol{y}|\boldsymbol{x};\theta)$, it is necessary to have supervised data for the task, which for many tasks cannot be found in large amounts. Prompt-based learning methods for NLP attempt to circumvent this issue by instead learning an LM that models the probability $P(\boldsymbol{x};\theta)$ of text $\boldsymbol{x}$ itself (details in §3) and using this probability to predict $\boldsymbol{y}$, reducing or obviating the need for large supervised datasets. In this section we lay out a mathematical description of the most fundamental form of prompting, which encompasses many works on prompting and can be expanded to cover others as well. Specifically, basic prompting predicts the highest-scoring $\hat{\boldsymbol{y}}$ in three steps.

Name	Notation	Example	Description
Input	$\boldsymbol{x}$	I love this movie.	One or multiple texts
Output	$\boldsymbol{y}$	++ (very positive)	Output label or text
Prompting Function	$f_{\text{prompt}}(\boldsymbol{x})$	`[X]` Overall, it was a `[Z]` movie.	A function that converts the input into a specific form by inserting the input $\boldsymbol{x}$ and adding a slot `[Z]` where answer $\boldsymbol{z}$ may be filled later.
Prompt	$\boldsymbol{x}'$	I love this movie. Overall, it was a `[Z]` movie.	A text where `[X]` is instantiated by input $\boldsymbol{x}$ but answer slot `[Z]` is not.
Filled Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$	I love this movie. Overall, it was a bad movie.	A prompt where slot `[Z]` is filled with any answer.
Answered Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}^*)$	I love this movie. Overall, it was a good movie.	A prompt where slot `[Z]` is filled with a true answer.
Answer	$\boldsymbol{z}$	"good", "fantastic", "boring"	A token, phrase, or sentence that fills `[Z]`

Table 2: Terminology and notation of prompting methods. $\boldsymbol{z}^*$ represents answers that correspond to true output $\boldsymbol{y}^*$.

2.2.1 Prompt Addition

In this step a *prompting function* $f_{\text{prompt}}(\cdot)$ is applied to modify the input text $\boldsymbol{x}$ into a *prompt* $\boldsymbol{x}' = f_{\text{prompt}}(\boldsymbol{x})$. In the majority of previous work (Kumar et al., 2016; McCann et al., 2018; Radford et al., 2019; Schick and Schütze, 2021a), this function consists of a two step process:

1. Apply a *template*, which is a textual string that has two slots: an *input slot* `[X]` for input $\boldsymbol{x}$ and an *answer slot* `[Z]` for an intermediate generated *answer* text $\boldsymbol{z}$ that will later be mapped into $\boldsymbol{y}$.
2. Fill slot `[X]` with the input text $\boldsymbol{x}$.

In the case of sentiment analysis where $\boldsymbol{x}$ ="I love this movie.", the template may take a form such as "`[X]` Overall, it was a `[Z]` movie.". Then, $\boldsymbol{x}'$ would become "I love this movie. Overall it was a `[Z]` movie." given the previous example. In the case of machine translation, the template may take a form such as "Finnish: `[X]` English: `[Z]`", where the text of the input and answer are connected together with headers indicating the language. We show more examples in Tab. 3

Notably, (1) the prompts above will have an empty slot to fill in for $\boldsymbol{z}$, either in the middle of the prompt or at the end. In the following text, we will refer to the first variety of prompt with a slot to fill in the middle of the text as a *cloze prompt*, and the second variety of prompt where the input text comes entirely before $\boldsymbol{z}$ as a *prefix prompt*. (2) In many cases these template words are not necessarily composed of natural language tokens; they could be virtual words (e.g. represented by numeric ids) which would be embedded in a continuous space later, and some prompting methods even generate continuous vectors directly (more in §4.3.2). (3) The number of `[X]` slots and the number of `[Z]` slots can be flexibly changed for the need of tasks at hand.

2.2.2 Answer Search

Next, we search for the highest-scoring text $\hat{\boldsymbol{z}}$ that maximizes the score of the LM. We first define $\mathcal{Z}$ as a set of permissible values for $\boldsymbol{z}$. $\mathcal{Z}$ could range from the entirety of the language in the case of generative tasks, or could be a small subset of the words in the language in the case of classification, such as defining $\mathcal{Z} = \{$"excellent", "good", "OK", "bad", "horrible"$\}$ to represent each of the classes in $\mathcal{Y} = \{$++, +, ~, -, --$\}$.

We then define a function $f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$ that fills in the location `[Z]` in prompt $\boldsymbol{x}'$ with the potential answer $\boldsymbol{z}$. We will call any prompt that has gone through this process as a *filled prompt*. Particularly, if the prompt is filled with a true answer, we will refer to it as an *answered prompt* (Tab. 2 shows an example). Finally, we search over the set of potential answers $\boldsymbol{z}$ by calculating the probability of their corresponding filled prompts using a pre-trained LM $P(\cdot; \theta)$

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z} \in \mathcal{Z}}{\text{search}}\, P(f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}); \theta). \tag{1}$$

This search function could be an *argmax* search that searches for the highest-scoring output, or *sampling* that randomly generates outputs following the probability distribution of the LM.

2.2.3 Answer Mapping

Finally, we would like to go from the highest-scoring *answer* $\hat{\boldsymbol{z}}$ to the highest-scoring *output* $\hat{\boldsymbol{y}}$. This is trivial in some cases, where the answer itself is the output (as in language generation tasks such as translation), but there

Type	Task	Input (`[X]`)	Template	Answer (`[Z]`)
Text CLS	Sentiment	I love this movie.	`[X]` The movie is `[Z]`.	great fantastic ...
	Topics	He prompted the LM.	`[X]` The text is about `[Z]`.	sports science ...
	Intention	What is taxi fare to Denver?	`[X]` The question is about `[Z]`.	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	`[X]` What about service? `[Z]`.	Bad Terrible ...
Text-pair CLS	NLI	`[X1]`: An old man with ... `[X2]`: A man walks ...	`[X1]`? `[Z]`, `[X2]`	Yes No ...
Tagging	NER	`[X1]`: Mike went to Paris. `[X2]`: Paris	`[X1][X2]` is a `[Z]` entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	`[X]` TL;DR: `[Z]`	The victim ... A woman ... ...
	Translation	Je vous aime.	French: `[X]` English: `[Z]`	I love you. I fancy you. ...

Table 3: Examples of *input*, *template*, and *answer* for different tasks. In the **Type** column, "CLS" is an abbreviation for "classification". In the **Task** column, "NLI" and "NER" are abbreviations for "natural language inference" (Bowman et al., 2015) and "named entity recognition" (Tjong Kim Sang and De Meulder, 2003) respectively.

are also other cases where multiple answers could result in the same output. For example, one may use multiple different sentiment-bearing words (e.g. "excellent", "fabulous", "wonderful") to represent a single class (e.g. "++"), in which case it is necessary to have a mapping between the searched answer and the output value.

2.3 Design Considerations for Prompting

Now that we have our basic mathematical formulation, we elaborate a few of the basic design considerations that go into a prompting method, which we will elaborate in the following sections:

- **Pre-trained Model Choice:** There are a wide variety of pre-trained LMs that could be used to calculate $P(\boldsymbol{x}; \theta)$. In §3 we give a primer on pre-trained LMs, specifically from the dimensions that are important for interpreting their utility in prompting methods.
- **Prompt Engineering:** Given that the prompt specifies the task, choosing a proper prompt has a large effect not only on the accuracy, but also on which task the model performs in the first place. In §4 we discuss methods to choose which prompt we should use as $f_{\text{prompt}}(\boldsymbol{x})$.
- **Answer Engineering:** Depending on the task, we may want to design $\mathcal{Z}$ differently, possibly along with the mapping function. In §5 we discuss different ways to do so.
- **Expanding the Paradigm:** As stated above, the above equations represent only the simplest of the various underlying frameworks that have been proposed to do this variety of prompting. In §6 we discuss ways to expand this underlying paradigm to further improve results or applicability.
- **Prompt-based Training Strategies:** There are also methods to train parameters, either of the prompt, the LM, or both. In §7, we summarize different strategies and detail their relative advantages.

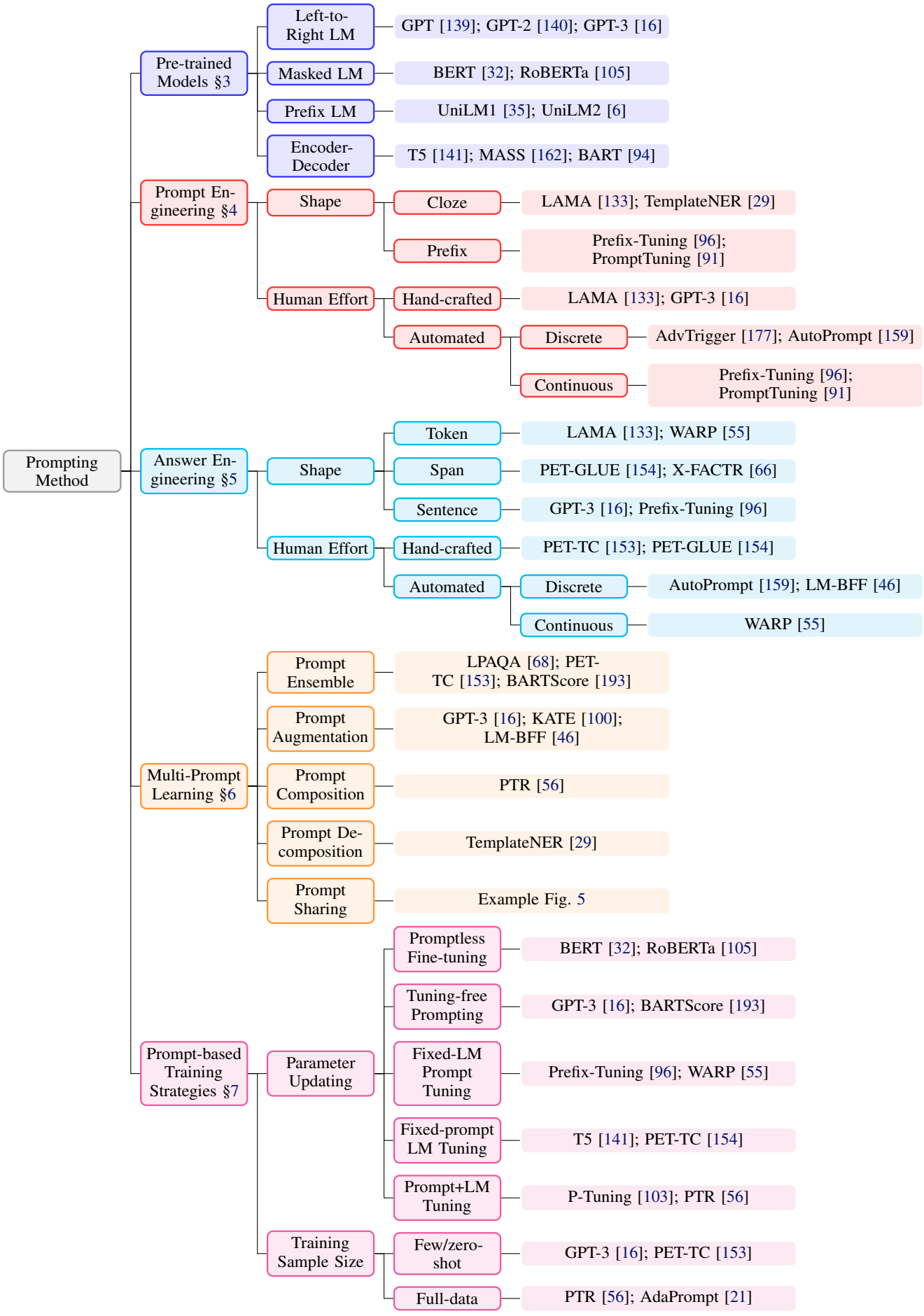


Figure 1: Typology of prompting methods.

3 Pre-trained Language Models

Given the large impact that pre-trained LMs have had on NLP in the pre-train and fine-tune paradigm, there are already a number of high-quality surveys that interested readers where interested readers can learn more (Raffel et al., 2020; Qiu et al., 2020; Xu et al., 2021; Doddapaneni et al., 2021). Nonetheless, in this chapter we present a systematic view of various pre-trained LMs which (i) organizes them along various axes in a more systematic way, (ii) particularly focuses on aspects salient to prompting methods. Below, we will detail them through the lens of *main training objective*, *type of text noising*, *auxiliary training objective*, *attention mask*, *typical architecture*, and *preferred application scenarios*. We describe each of these objectives below, and also summarize a number of pre-trained LMs along each of these axes in Tab. 13 in the appendix.

3.1 Training Objectives

The main training objective of a pre-trained LM almost invariably consists of some sort of objective predicting the probability of text $\boldsymbol{x}$.

Standard Language Model (SLM) objectives do precisely this, training the model to optimize the probability $P(\boldsymbol{x})$ of text from a training corpus (Radford et al., 2019). In these cases, the text is generally predicted in an *autoregressive* fashion, predicting the tokens in the sequence one at a time. This is usually done from left to right (as detailed below), but can be done in other orders as well.

A popular alternative to standard LM objectives are *denoising* objectives, which apply some noising function $\tilde{\boldsymbol{x}} = f_{\text{noise}}(\boldsymbol{x})$ to the input sentence (details in the following subsection), then try to predict the original input sentence given this noised text $P(\boldsymbol{x}|\tilde{\boldsymbol{x}})$. There are two common flavors of these objectives:

Corrupted Text Reconstruction (CTR) These objectives restore the processed text to its uncorrupted state by calculating loss over *only* the noised parts of the input sentence.

Full Text Reconstruction (FTR) These objectives reconstruct the text by calculating the loss over the *entirety* of the input texts whether it has been noised or not (Lewis et al., 2020a).

The main training objective of the pre-trained LMs plays an important role in determining its applicability to particular prompting tasks. For example, left-to-right autoregressive LMs may be particularly suitable for prefix prompts, whereas reconstruction objectives may be more suitable for cloze prompts. In addition, models trained with standard LM and FTR objectives may be more suitable for tasks regarding text generation, whereas other tasks such as classification can be formulated using models trained with any of these objectives.

In addition to the main training objectives above, a number of *auxiliary objectives* have been engineered to further improve models' ability to perform certain varieties of downstream tasks. We list some commonly-used auxiliary objectives in Appendix A.2.

3.2 Noising Functions

In training objectives based on reconstruction, the specific type of corruption applied to obtain the noised text $\tilde{\boldsymbol{x}}$ has an effect on the efficacy of the learning algorithm. In addition, prior knowledge can be incorporated by controlling the type of noise, e.g. the noise could focus on entities of a sentence, which allows us to learn a pre-trained model with particularly high predictive performance for entities. In the following, we introduce several types of noising functions, and give detailed examples in Tab. 4.

Operation	Element	Original Text	Corrupted Text
Mask	one token	Jane will move to New York .	Jane will `[Z]` to New York .
	two tokens	Jane will move to New York .	Jane will `[Z]` `[Z]` New York .
	one entity	Jane will move to New York .	Jane will move to `[Z]` .
Replace	one token	Jane will move to New York .	Jane will move `[X]` New York .
	two tokens	Jane will move to New York .	Jane will move `[X]` `[Y]` York .
	one entity	Jane will move to New York .	Jane will move to `[X]` .
Delete	one token	Jane ~~will~~ move to New York .	Jane move to New York .
	two token	Jane ~~will move~~ to New York .	Jane to New York .
Permute	token	Jane will move to New York .	New York . Jane will move to
Rotate	none	Jane will move to New York .	to New York . Jane will move
Concatenation	two languages	Jane will move to New York .	Jane will move to New York . `[/s]` 简将搬到纽约。

Table 4: Detailed examples for different noising operations.

Masking (e.g. Devlin et al. (2019)) The text will be masked in different levels, replacing a token or multi-token span with a special token such as `[MASK]`. Notably, masking can either be random from some distribution or specifically designed to introduce prior knowledge, such as the above-mentioned example of masking entities to encourage the model to be good at predicting entities.

Replacement (e.g. Raffel et al. (2020)) Replacement is similar to masking, except that the token or multi-token span is not replaced with a `[MASK]` but rather another token or piece of information (e.g., an image region (Su et al., 2020)).

Deletion (e.g. Lewis et al. (2020a)) Tokens or multi-token spans will be deleted from a text without the addition of `[MASK]` or any other token. This operation is usually used together with the FTR loss.

Permutation (e.g. Liu et al. (2020a)) The text is first divided into different spans (tokens, sub-sentential spans, or sentences), and then these spans are be permuted into a new text.

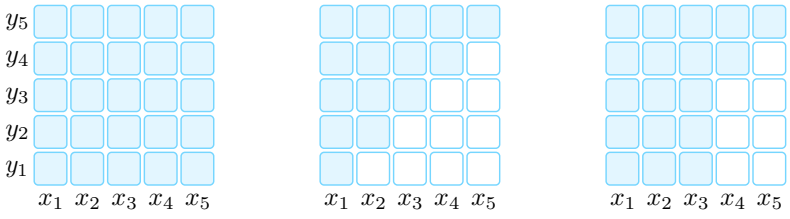
3.3 Directionality of Representations

A final important factor that should be considered in understanding pre-trained LMs and the difference between them is the directionality of the calculation of representations. In general, there are two widely used ways to calculate such representations:

Left-to-Right The representation of each word is calculated based on the word itself and all previous words in the sentence. For example, if we have a sentence "This is a good movie", the representation of the word "good" would be calculated based on previous words. This variety of factorization is particularly widely used when calculating standard LM objectives or when calculating the output side of an FTR objective, as we discuss in more detail below.

Bidirectional The representation of each word is calculated based on all words in the sentence, including words to the left of the current word. In the example above, "good" would be influenced by all words in the sentence, even the following "movie".

In addition to the two most common directionalities above, it is also possible to mix the two strategies together in a single model (Dong et al., 2019; Bao et al., 2020), or perform conditioning of the representations in a randomly permuted order (Yang et al., 2019), although these strategies are less widely used. Notably, when implementing these strategies within a neural model, this conditioning is generally implemented through *attention masking*, which masks out the values in an attentional model (Bahdanau et al., 2014), such as the popular Transformer architecture (Vaswani et al., 2017). Some examples of such attention masks are shown in Figure 2.



(a) Full. (b) Diagonal. (c) Mixture.

Figure 2: Three popular attention mask patterns, where the subscript t indicates the t-th timestep. A shaded box at (i, j) indicates that the attention mechanism is allowed to attend to the input element i at output time step j. A white box indicates that the attention mechanism is not allowed to attend to the corresponding i and j combination.

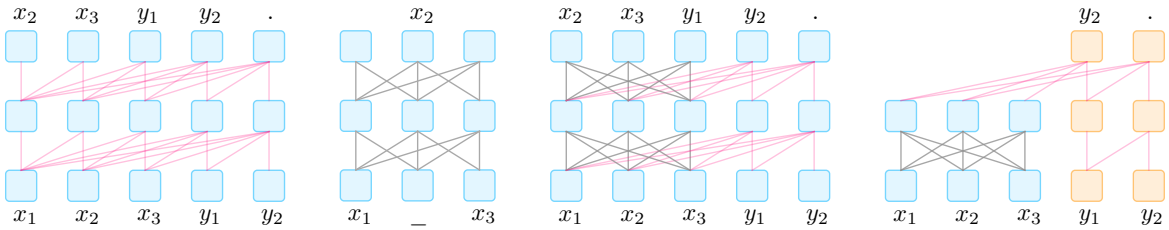
3.4 Typical Pre-training Methods

With the above concepts in mind, we introduce four popular pre-training methods, resulting from diverse combinations of objective, noising function, and directionality. These are described below, and summarized in Fig. 3 and Tab. 5.

3.4.1 Left-to-Right Language Model

Left-to-right LMs (L2R LMs), a variety of *auto-regressive LM*, predict the upcoming words or assign a probability $P(\boldsymbol{x})$ to a sequence of words $\boldsymbol{x} = x_1, \cdots, x_n$ (Jurafsky and Martin, 2021). The probability is commonly broken down using the chain rule in a left-to-right fashion: $P(\boldsymbol{x}) = P(x_1) \times \cdots P(x_n|x_1 \cdots x_{n-1})$.[3]

[3]Similarly, a right-to-left LM can predict preceding words based on the future context, such as $P(x_i|x_{i+1}, \cdots, x_n)$.



(a) Left-to-right LM. (b) Masked LM. (c) Prefix LM. (d) Encoder-Decoder.

Figure 3: Typical paradigms of pre-trained LMs.

Example & Applicable Scenario

Left-to-right LMs have been standard since their proposal by Markov in 1913 (Markov, 2006), and have been used continuously since then in both count-based (Goodman, 2001) and neural forms (Bengio et al., 2003; Mikolov et al., 2010; Radford and Narasimhan, 2018). Representative examples of modern pre-trained left-to-right LMs include GPT-3 (Brown et al., 2020), and GPT-Neo (Black et al., 2021).
L2R pre-trained LMs are also the popular backbone that many prompting methods adopt (Radford et al., 2019; Brown et al., 2020) . One practical reason for this is that many such models are large (PanGu-α (Zeng et al., 2021), Ernie-3 (Sun et al., 2021)) and ponderous to train, or not even available publicly. Thus using these models in the pre-train and fine-tune regimen is often not possible.

LMs	$\boldsymbol{x}$			$\boldsymbol{y}$			Application
	Mask	Noise	Main Obj.	Mask	Noise	Main Obj.	
L2R	Diagonal	None	SLM	-	-	-	NLU & NLG
Mask	Full	Mask	CTR	-	-	-	NLU
Prefix	Full	Any	CTR	Diagonal	None	SLM	NLU & NLG
En-De	Full	Any	None†	Diagonal	None	FTR/CRT	NLU & NLG

Table 5: Typical architectures for pre-trained LMs. $\boldsymbol{x}$ and $\boldsymbol{y}$ represent text to be encoded and decoded, respectively. **SLM**: Standard language model. **CTR**: Corrupted text reconstruction. **FTR**: Full text reconstruction. †: Encoder-decoder architectures usually apply objective functions to the decoder only.

3.4.2 Masked Language Models

While autoregressive language models provide a powerful tool for modeling the probability of text, they also have disadvantages such as requiring representations be calculated from left-to-right. When the focus is shifted to generating the optimal representations for down-stream tasks such as classification, many other options become possible, and often preferable. One popular bidirectional objective function used widely in representation learning is the *masked language model* (MLM; Devlin et al. (2019)), which aims to predict masked text pieces based on surrounded context. For example, $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represents the probability of the word x_i given the surrounding context.

Example & Applicable Scenario

Representative pre-trained models using MLMs include: BERT (Devlin et al., 2019), ERNIE (Zhang et al., 2019; Sun et al., 2019b) and many variants. In prompting methods, MLMs are generally most suitable for natural language understanding or analysis tasks (e.g., text classification, natural language inference , and extractive question answering). These tasks are often relatively easy to be reformulated into cloze problems, which are consistent with the training objectives of the MLM. Additionally, MLMs have been a pre-trained model of choice when exploring methods that combine prompting with fine-tuning, elaborated further in §7.

3.4.3 Prefix and Encoder-Decoder

For conditional text generation tasks such as translation and summarization where an input text $\boldsymbol{x} = x_1, \cdots, x_n$ is given and the goal is to generate target text $\boldsymbol{y}$, we need a pre-trained model that is both capable of encoding the input text and generating the output text. There are two popular architectures for this purpose that share a common

thread of (1) using an encoder with fully-connected mask to encode the source $\boldsymbol{x}$ first and then (2) decode the target $\boldsymbol{y}$ auto-regressively (from the left to right).

Prefix Language Model The prefix LM is a left-to-right LM that decodes $\boldsymbol{y}$ conditioned on a prefixed sequence $\boldsymbol{x}$, which is encoded by the *same* model parameters but with a fully-connected mask. Notably, to encourage the prefix LM to learn better representations of the input, a corrupted text reconstruction objective is usually applied over $\boldsymbol{x}$, in addition to a standard conditional language modeling objective over $\boldsymbol{y}$.

Encoder-decoder The encoder-decoder model is a model that uses a left-to-right LM to decode $\boldsymbol{y}$ conditioned on a *separate* encoder for text $\boldsymbol{x}$ with a fully-connected mask; the parameters of the encoder and decoder are not shared. Similarly to the prefix LM, diverse types of noising can be applied to the input $\boldsymbol{x}$.

Example & Applicable Scenario

Prefix LMs have been explored in UniLM 1-2 (Dong et al., 2019; Bao et al., 2020) and ERNIE-M (Ouyang et al., 2020) while encoder-decoder models are widely used in pre-trained models such as T5 (Raffel et al., 2020), BART (Lewis et al., 2020a), MASS (Song et al., 2019) and their variants.
Pre-trained models with prefix LMs and encoder-decoder paradigms can be naturally used to text generation tasks with (Dou et al., 2021) or without (Yuan et al., 2021a; Liu and Liu, 2021) prompting using input texts. However, recent studies reveal that other non-generation tasks, such as information extraction (Cui et al., 2021), question answering (Khashabi et al., 2020) , and text generation evaluation (Yuan et al., 2021b) can be reformulated a generation problems by providing appropriate prompts. Therefore, prompting methods (i) broaden the applicability of these generation-oriented pre-trained models. For example, pre-trained models like BART are less used in NER while prompting methods make BART applicable, and (ii) breaks the difficulty of unified modelling among different tasks (Khashabi et al., 2020).

4 Prompt Engineering

Prompt engineering is the process of creating a prompting function $f_{\text{prompt}}(\boldsymbol{x})$ that results in the most effective performance on the downstream task. In many previous works, this has involved *prompt template engineering*, where a human engineer or algorithm searches for the best template for each task the model is expected to perform. As shown in the "Prompt Engineering" section of Fig.1, one must first consider the *prompt shape*, and then decide whether to take a *manual* or *automated* approach to create prompts of the desired shape, as detailed below.

4.1 Prompt Shape

As noted above, there are two main varieties of prompts: *cloze prompts* (Petroni et al., 2019; Cui et al., 2021), which fill in the blanks of a textual string, and *prefix prompts* (Li and Liang, 2021; Lester et al., 2021), which continue a string prefix. Which one is chosen will depend both on the task and the model that is being used to solve the task. In general, for tasks regarding generation, or tasks being solved using a standard auto-regressive LM, prefix prompts tend to be more conducive, as they mesh well with the left-to-right nature of the model. For tasks that are solved using masked LMs, cloze prompts are a good fit, as they very closely match the form of the pre-training task. Full text reconstruction models are more versatile, and can be used with either cloze or prefix prompts. Finally, for some tasks regarding multiple inputs such as *text pair classification*, prompt templates must contain space for two inputs, `[X1]` and `[X2]`, or more.

4.2 Manual Template Engineering

Perhaps the most natural way to create prompts is to manually create intuitive templates based on human introspection. For example, the seminal LAMA dataset (Petroni et al., 2019) provides manually created cloze templates to probe knowledge in LMs. Brown et al. (2020) create manually crafted prefix prompts to handle a wide variety of tasks, including question answering, translation, and probing tasks for common sense reasoning. Schick and Schütze (2020, 2021a,b) use pre-defined templates in a few-shot learning setting on text classification and conditional text generation tasks.

4.3 Automated Template Learning

While the strategy of manually crafting templates is intuitive and does allow solving various tasks with some degree of accuracy, there are also several issues with this approach: (1) creating and experimenting with these prompts is an art that takes time and experience, particularly for some complicated tasks such as semantic parsing (Shin et al., 2021); (2) even experienced prompt designers may fail to manually discover optimal prompts (Jiang et al., 2020c).

To address these problems, a number of methods have been proposed to automate the template design process. In particular, the automatically induced prompts can be further separated into *discrete prompts*, where the prompt is an

actual text string, and *continuous prompts*, where the prompt is instead described directly in the embedding space of the underlying LM.

One other orthogonal design consideration is whether the prompting function $f_{\text{prompt}}(\boldsymbol{x})$ is *static*, using essentially the same prompt template for each input, or *dynamic*, generating a custom template for each input. Both static and dynamic strategies have been used for different varieties of discrete and continuous prompts, as we will mention below.

4.3.1 Discrete Prompts

Works on discovering *discrete prompts* (a.k.a *hard prompts*) automatically search for templates described in a discrete space, usually corresponding to natural language phrases. We detail several methods that have been proposed for this below:

D1: Prompt Mining Jiang et al. (2020c)'s MINE approach is a mining-based method to automatically find templates given a set of training inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$. This method scrapes a large text corpus (e.g. Wikipedia) for strings containing $\boldsymbol{x}$ and $\boldsymbol{y}$, and finds either the *middle words* or *dependency paths* between the inputs and outputs. Frequent middle words or dependency paths can serve as a template as in "`[X]` middle words `[Z]`".

D2: Prompt Paraphrasing Paraphrasing-based approaches take in an existing seed prompt (e.g. manually constructed or mined), and paraphrases it into a set of other candidate prompts, then selects the one that achieves the highest training accuracy on the target task. This paraphrasing can be done in a number of ways, including using round-trip translation of the prompt into another language then back (Jiang et al., 2020c), using replacement of phrases from a thesaurus (Yuan et al., 2021b), or using a neural prompt rewriter specifically optimized to improve accuracy of systems using the prompt (Haviv et al., 2021). Notably, Haviv et al. (2021) perform paraphrasing *after* the input $\boldsymbol{x}$ is input into the prompt template, allowing a different paraphrase to be generated for each individual input.

D3: Gradient-based Search Wallace et al. (2019a) applied a gradient-based search over actual tokens to find short sequences that can trigger the underlying pre-trained LM to generate the desired target prediction. This search is done in an iterative fashion, stepping through tokens in the prompt . Built upon this method, Shin et al. (2020) automatically search for template tokens using downstream application training samples and demonstrates strong performance in prompting scenarios.

D4: Prompt Generation Other works treat the generation of prompts as a text generation task and use standard natural language generation models to perform this task. For example, Gao et al. (2021) introduce the seq2seq pre-trained model T5 into the template search process. Since T5 has been pre-trained on a task of filling in missing spans, they use T5 to generate template tokens by (1) specifying the position to insert template tokens within a template[4] (2) provide training samples for T5 to decode template tokens. Ben-David et al. (2021) propose a domain adaptation algorithm that trains T5 to generate unique domain relevant features (DRFs; a set of keywords that characterize domain information) for each input. Then those DRFs can be concatenated with the input to form a template and be further used by downstream tasks.

D5: Prompt Scoring Davison et al. (2019) investigate the task of knowledge base completion and design a template for an input (head-relation-tail triple) using LMs. They first hand-craft a set of templates as potential candidates, and fill the input and answer slots to form a filled prompt. They then use a unidirectional LM to score those filled prompts, selecting the one with the highest LM probability. This will result in custom template for each individual input.

4.3.2 Continuous Prompts

Because the purpose of prompt construction is to find a method that allows an LM to effectively perform a task, rather than being for human consumption, it is not necessary to limit the prompt to human-interpretable natural language. Because of this, there are also methods that examine *continuous prompts* (a.k.a. *soft prompts*) that perform prompting directly in the embedding space of the model. Specifically, continuous prompts remove two constraints: (1) relax the constraint that the embeddings of template words be the embeddings of natural language (e.g., English) words. (2) Remove the restriction that the template is parameterized by the pre-trained LM's parameters. Instead, templates have their own parameters that can be tuned based on training data from the downstream task. We highlight several representative methods below.

[4]The number of template tokens do not need to be pre-specified since T5 can decode multiple tokens at a masked position.

C1: Prefix Tuning Prefix Tuning (Li and Liang, 2021) is a method that prepends a sequence of continuous task-specific vectors to the input, while keeping the LM parameters frozen. Mathematically, this consists of optimizing over the following log-likelihood objective given a trainable prefix matrix M_ϕ and a fixed pre-trained LM parameterized by θ.

$$\max_{\phi} \log P(\boldsymbol{y}|\boldsymbol{x}; \theta; \phi) = \max_{\phi} \sum_{y_i} \log P(y_i|h_{<i}; \theta; \phi) \tag{2}$$

In Eq. 2, $h_{<i} = [h_{<i}^{(1)}; \cdots; h_{<i}^{(n)}]$ is the concatenation of all neural network layers at time step i. It is copied from M_ϕ directly if the corresponding time step is within the prefix (h_i is $M_\phi[i]$), otherwise it is computed using the pre-trained LM.

Experimentally, Li and Liang (2021) observe that such continuous prefix-based learning is more sensitive to different initialization in low-data settings than the use of discrete prompts with real words. Similarly, Lester et al. (2021) prepend the input sequence with special tokens to form a template and tune the embeddings of these tokens directly. Compared to Li and Liang (2021)'s method, this adds fewer parameters as it doesn't introduce additional tunable parameters within each network layer. Tsimpoukelli et al. (2021) train a vision encoder that encodes an image into a sequence of embeddings that can be used to prompt a frozen auto-regressive LM to generate the appropriate caption. They show that the resulting model can perform few-shot learning for vision-language tasks such as visual question answering etc. Different from the above two works, the prefix used in (Tsimpoukelli et al., 2021) is sample-dependent, namely a representation of input images, instead of a task embedding.

C2: Tuning Initialized with Discrete Prompts There are also methods that initialize the search for a continuous prompt using a prompt that has already been created or discovered using discrete prompt search methods. For example, Zhong et al. (2021b) first define a template using a discrete search method such as AUTOPROMPT (Shin et al., 2020)'s, initialize virtual tokens based on this discovered prompt, then fine-tune the embeddings to increase task accuracy. This work found that initializing with manual templates can provide a better starting point for the search process. Qin and Eisner (2021) propose to learn a mixture of soft templates for each input where the weights and parameters for each template are jointly learned using training samples. The initial set of templates they use are either manually crafted ones or those obtained using the "prompt mining" method. Similarly, Hambardzumyan et al. (2021) introduce the use of a continuous template whose shape follows a manual prompt template.

C3: Hard-Soft Prompt Hybrid Tuning Instead of using a purely learnable prompt template, these methods insert some tunable embeddings into a hard prompt template. Liu et al. (2021b) propose "P-tuning", where continuous prompts are learned by inserting trainable variables into the embedded input. To account for interaction between prompt tokens, they represent prompt embeddings as the output of a BiLSTM (Graves et al., 2013). P-tuning also introduces the use of task-related anchor tokens (such as "capital" in relation extraction) within the template for further improvement. These anchor tokens are not tuned during training. Han et al. (2021) propose prompt tuning with rules (PTR), which uses manually crafted sub-templates to compose a complete template using logic rules. To enhance the representation ability of the resulting template, they also insert several virtual tokens whose embeddings can be tuned together with the pre-trained LMs parameters using training samples. The template tokens in PTR contain both actual tokens and virtual tokens. Experiment results demonstrate the effectiveness of this prompt design method in relation classification tasks.

5 Answer Engineering

In contrast to prompt engineering, which designs appropriate inputs for prompting methods, *answer engineering* aims to search for an answer space $\mathcal{Z}$ and a map to the original output $\mathcal{Y}$ that results in an effective predictive model. Fig.1's "Answer Engineering" section illustrates two dimensions that must be considered when performing answer engineering: deciding the *answer shape* and choosing an *answer design method*.

5.1 Answer Shape

The shape of an answer characterizes its granularity. Some common choices include:

- **Tokens:** One of the tokens in the pre-trained LM's vocabulary, or a subset of the vocabulary.
- **Span:** A short multi-token span. These are usually used together with cloze prompts.
- **Sentence:** A sentence or document. These are commonly used with prefix prompts.

In practice, how to choose the shape of acceptable answers depends on the task we want to perform. Token or text-span answer spaces are widely used in classification tasks (e.g. sentiment classification; Yin et al. (2019)), but also other tasks such as relation extraction (Petroni et al., 2019) or named entity recognition (Cui et al., 2021). Longer phrasal or sentential answers are often used in language generation tasks (Radford et al., 2019), but also

used in other tasks such as multiple-choice question answering (where the scores of multiple phrases are compared against each-other; Khashabi et al. (2020)).

5.2 Answer Space Design Methods

The next question to answer is how to design the appropriate answer space $\mathcal{Z}$, as well as the mapping to the output space $\mathcal{Y}$ if the answers are not used as the final outputs.

5.2.1 Manual Design

In manual design, the space of potential answers $\mathcal{Z}$ and its mapping to $\mathcal{Y}$ are crafted manually by an interested system or benchmark designer. There are a number of strategies that can be taken to perform this design.

Unconstrained Spaces In many cases, the answer space $\mathcal{Z}$ is the space of all tokens (Petroni et al., 2019), fixed-length spans (Jiang et al., 2020a), or token sequences (Radford et al., 2019). In these cases, it is most common to directly map answer $\boldsymbol{z}$ to the final output $\boldsymbol{y}$ using the identity mapping.

Constrained Spaces However, there are also cases where the space of possible outputs is constrained. This is often performed for tasks with a limited label space such as text classification or entity recognition, or multiple-choice question answering. To give some examples, Yin et al. (2019) manually design lists of words relating to relevant topics ("health", "finance", "politics", "sports", etc.), emotions ("anger", "joy", "sadness", "fear", etc.), or other aspects of the input text to be classified. Cui et al. (2021) manually design lists such as "person", "location", etc. for NER tasks. In these cases, it is necessary to have a mapping between the answer $\mathcal{Z}$ and the underlying class $\mathcal{Y}$.

With regards to multiple-choice question answering, it is common to use an LM to calculate the probability of an output among multiple choices, with Zweig et al. (2012) being an early example.

5.2.2 Discrete Answer Search

As with manually created prompts, it is possible that manually created answers are sub-optimal for getting the LM to achieve ideal prediction performance. Because of this, there is some work on automatic answer search, albeit less than that on searching for ideal prompts. These work on both discrete answer spaces (this section) and continuous answer spaces (the following).

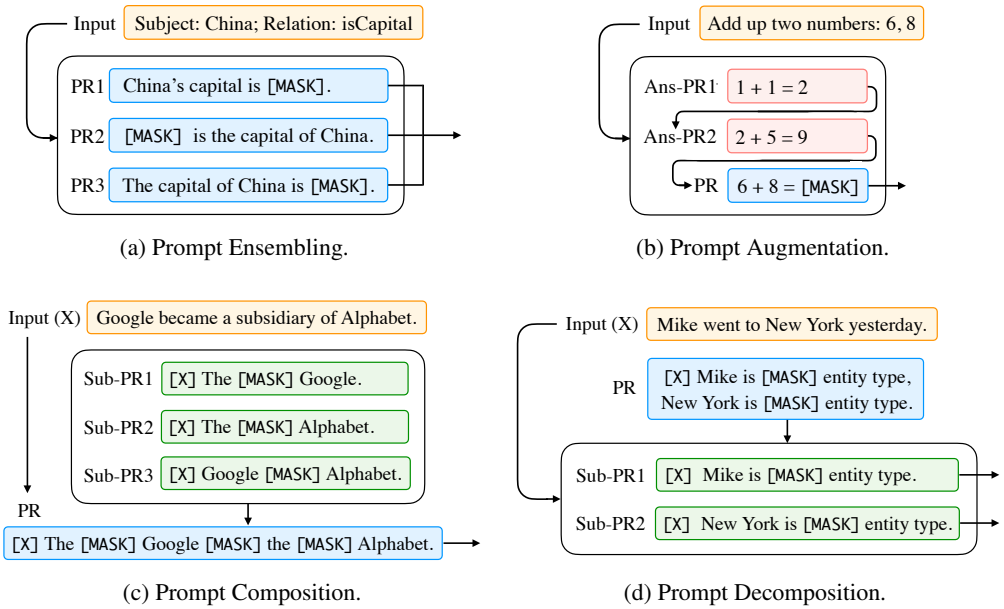
Answer Paraphrasing These methods start with an initial answer space $\mathcal{Z}'$, and then use paraphrasing to expand this answer space to broaden its coverage (Jiang et al., 2020b). Given a pair of answer and output $\langle \boldsymbol{z}', \boldsymbol{y} \rangle$, we define a function that generates a paraphrased set of answers $\text{para}(\boldsymbol{z}')$. The probability of the final output is then defined as the marginal probability *all* of the answers in this paraphrase set $P(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{z} \in \text{para}(\boldsymbol{z}')} P(\boldsymbol{z}|\boldsymbol{x})$. This paraphrasing can be performed using any method, but Jiang et al. (2020b) specifically use a back-translation method, first translating into another language then back to generate a list of multiple paraphrased answers.

Prune-then-Search In these methods, first, an initial pruned answer space of several plausible answers $\mathcal{Z}'$ is generated, and then an algorithm further searches over this pruned space to select a final set of answers. Note that in some of the papers introduced below, they define a function from label $\boldsymbol{y}$ to a single answer token $\boldsymbol{z}$, which is often called a *verbalizer* (Schick and Schütze, 2021a). Schick and Schütze (2021a); Schick et al. (2020) find tokens containing at least two alphabetic characters that are frequent in a large unlabeled dataset. In the search step, they iteratively compute a word's suitability as a representative answer $\boldsymbol{z}$ for a label $\boldsymbol{y}$ by maximizing the likelihood of the label over training data. Shin et al. (2020) learn a logistic classifier using the contextualized representation of the `[Z]` token as input. In the search step, they select the top-k tokens that achieve the highest probability score using the learned logistic classifier in the first step. Those selected tokens will form the answer. Gao et al. (2021) first construct a pruned search space $\mathcal{Z}'$ by selecting top-k vocabulary words based on their generation probability at the `[Z]` position determined by training samples. Then the search space is further pruned down by only selecting a subset of words within $\mathcal{Z}'$ based on their zero-shot accuracy on the training samples. (2) In the search step, they fine-tune the LM with fixed templates together with every answer mapping using training data and select the best label word as the answer based on the accuracy on the development set.

Label Decomposition When performing relation extraction, Chen et al. (2021b) automatically decompose each relation label into its constituent words and use them as an answer. For example, for the relation `per:city_of_death`, the decomposed label words would be {`person`, `city`, `death`}. The probability of the answer span will be calculated as the sum of each token's probability.

5.2.3 Continuous Answer Search

Very few works explore the possibility of using soft answer tokens which can be optimized through gradient descent. Hambardzumyan et al. (2021) assign a virtual token for each class label and optimize the token embedding for each



(a) Prompt Ensembling.

(b) Prompt Augmentation.

(c) Prompt Composition.

(d) Prompt Decomposition.

Figure 4: Different multi-prompt learning strategies. We use different colors to differentiate different components as follows. " " for input text, " " for prompt, " " for answered prompt. " " for sub-prompt. We use the following abbreviations. "PR" for prompt, "Ans-PR" for answered prompt, "Sub-PR" for sub-prompt.

class together with prompt token embeddings. Since the answer tokens are optimized directly in the embedding space, they do not make use of the embeddings learned by the LM and instead learn an embedding from scratch for each label.

6 Multi-Prompt Learning

The prompt engineering methods we discussed so far focused mainly on constructing a *single* prompt for an input. However, a significant body of research has demonstrated that the use of multiple prompts can further improve the efficacy of prompting methods, and we will call these methods *multi-prompt learning* methods. In practice, there are several ways to extend the single prompt learning to the use multiple prompts, which have a variety of motivations. We summarize representative methods in the "Multi-prompt Learning" section of Fig.1 as well as Fig.4.

6.1 Prompt Ensembling

Prompt ensembling is the process of using multiple *unanswered* prompts for an input at inference time to make predictions. An example is shown in Fig. 4-(a). The multiple prompts can either be discrete prompts or continuous prompts.[5] This sort of prompt ensembling can (1) leverage the complementary advantages of different prompts, (2) alleviate the cost of prompt engineering, since choosing one best-performing prompt is challenging, (3) stabilize performance on downstream tasks.

Prompt ensembling is connected to ensembling methods that are used to combine together multiple systems, which have a long history in machine learning (Ting and Witten, 1997; Zhou et al., 2002; Duh et al., 2011). Current research also borrows ideas from these works to derive effective ways for prompt ensembling, as described below.

Uniform averaging The most intuitive way to combine the predictions when using multiple prompts is to take the average of probabilities from different prompts. Concretely, this indicates that $P(\boldsymbol{z}|\boldsymbol{x}) \coloneqq \frac{1}{K}\sum_i^K P(\boldsymbol{z}|f_{\text{prompt},i}(\boldsymbol{x}))$ where $f_{\text{prompt},i}(\cdot)$ is the ith prompt in the prompt ensemble. Jiang et al. (2020c) first filter their prompts by selecting K prompts that achieve the highest accuracy on the training set, and then use the average log probabilities obtained from the top K prompts to calculate the probability for a single token at `[Z]` position when performing factual probing tasks. Schick and Schütze (2021a) also try a simple average when using an ensemble model to annotate an unlabeled dataset. When performing text generation evaluation, Yuan et al. (2021b) formulates this task as a text generation problem and take the average of the final generation scores obtained using different prompts.

Weighted averaging Simple uniform averaging of results from multiple prompts is easy to implement, but can also be suboptimal given that some prompts are more performant than others. To account for this, some works also

[5]Multiple continuous prompts are typically learned by using different initializations or different random seeds.

explore to use of weighted averages for prompt ensembling where each prompt is associated with a weight. The weights are typically pre-specified based on prompt performance or optimized using a training set. For example, Jiang et al. (2020c) learn the weight for each prompt by maximizing the probability of the target output over training data. Qin and Eisner (2021) use the same approach except that the weight for each prompt is optimized together with soft prompt parameters. Besides, Qin and Eisner (2021) also introduce a data-dependent weighting strategy where the probability of the input appearing in that prompt is considered in weighting different prompts as well. Schick and Schütze (2021a,b) set the weight for each prompt proportional to the accuracy on the training set before training.

Majority voting For classification tasks, majority voting can also be used to combine the results from different prompts (Lester et al., 2021; Hambardzumyan et al., 2021).

Knowledge distillation An ensemble of deep learning models can typically improve the performance, and this superior performance can be distilled into a single model using knowledge distillation (Allen-Zhu and Li, 2020). To incorporate this idea, Schick and Schütze (2021a,b, 2020) train a separate model for each manually-created template-answer pair, and use the ensemble of them to annotate an unlabeled dataset. Then the final model is trained to distill the knowledge from the annotated dataset. Gao et al. (2021) use a similar ensemble method on their automatically generated templates.

Prompt ensembling for text generation There is relatively little work on prompt ensembling for generation tasks (i.e. tasks where the answers is a string of tokens instead of a single one). A simple way to perform ensembling in this case is to use standard methods that generate the output based on the ensembled probability of the next word in the answer sequence $P(z_t|\boldsymbol{x}, z_{<t}) \coloneqq \frac{1}{K}\sum_i^K P(z_t|f_{\text{prompt},i}(\boldsymbol{x}), z_{<t})$. In contrast, Schick and Schütze (2020) train a separate model for each prompt $f_{\text{prompt},i}(\boldsymbol{x})$, and thus storing each of these fine-tuned LMs in memory is infeasible. Instead, they first decode generations using each model and then score each generation by averaging their generation probability across all models.

6.2 Prompt Augmentation

Prompt augmentation, also sometimes called *demonstration learning* (Gao et al., 2021), provides a few additional *answered prompts* that can be used to demonstrate how the LM should provide the answer to the actual prompt instantiated with the input $\boldsymbol{x}$. For example, instead of just providing a prompt of "China's capital is `[Z]` .", the prompt can be prefaced by a few examples such as "Great Britain's capital is London . Japan's capital is Tokyo . China's capital is `[Z]` ." Another example of performing addition of two numbers can be found in Fig. 4-(b). These few-shot demonstrations take advantage of the ability of strong language models to learn repetitive patterns (Brown et al., 2020).

Although the idea of prompt augmentation is simple, there are several aspects that make it challenging: (1) *Sample Selection:* how to choose the most effective examples? (2) *Sample Ordering:* How to order the chosen examples with the prompt?

Sample Selection Researchers have found that the choice of examples used in this few-shot scenario can result in very different performance, ranging from near state-of-the-art accuracy on some tasks to near random guess (Lu et al., 2021). To address this issue, Gao et al. (2021); Liu et al. (2021a) utilize sentence embeddings to sample examples that are close to the input in this embedding space. To measure the generalization capability of pre-trained LMs to perform new tasks based on instructions, Mishra et al. (2021) provide both positive samples and negative samples that highlight things to avoid.

Sample Ordering Lu et al. (2021) found that the order of answered prompts provided to the model plays an important role in model performance, and propose entropy-based methods to score different candidate permutations. Kumar and Talukdar (2021) search for a good permutation of training examples as augmented prompts and learn a separator token between the prompts for further gains in performance.

Prompt augmentation is closely related to retrieval-based methods that provide more textual context to the model to improve performance (Guu et al., 2018), a method which has also been shown to be effective in prompt-based learning (Petroni et al., 2020). However, the key difference lies in the fact that prompt augmentation also leverages the template and answer, while larger context learning does not.

6.3 Prompt Composition

For those composable tasks, which can be composed based on more fundamental subtasks, we can also perform *prompt composition*, using multiple sub-prompts, each for one subtask, and then defining a composite prompt based on those sub-prompts. This process is illustrated in Fig. 4-(c). For example, in the relation extraction task, which aims to extract the relation of two entities, we can break down the task into several subtasks including identifying the characteristics of entities and classifying the relationships between entities. Based on this intuition, Han et al.

(2021) first use multiple manually created sub-prompts for entity recognition and relation classification and then compose them into a complete prompt based on logic rules for relation extraction.

6.4 Prompt Decomposition

For tasks where multiple predictions should be performed for one sample (e.g., sequence labeling), directly defining a holistic prompt with regards to the entire input text $\boldsymbol{x}$ is challenging. One intuitive method to address this problem is to break down the holistic prompt into different sub-prompts, and then answer each sub-prompt separately. Fig.4-(d) illustrates this idea with an example from the named entity recognition task, which aims to identify all named entities in an input sentence. In this case, the input will first be converted into a set of text spans, and the model can then be prompted to predict the entity type (including "Not an Entity") for each span. It is not easy to predict all the span types at the same time due to the large number of spans, so different prompts for each span can be created and predicted separately. This sort of *prompt decomposition* for named entity recognition has been explored by Cui et al. (2021) where they apply the approach we discussed here.

7 Training Strategies for Prompting Methods

With the methods in the above sections, it is now clear how to obtain an appropriate prompt (or prompts) and corresponding answers. Now we discuss about methods that explicitly train models in concert with prompting methods, as outlined in the "Training Strategies" section of Fig.1.

7.1 Training Settings

In many cases, prompting methods can be used without *any* explicit training of the LM for the down-stream task, simply taking an LM that has been trained to predict the probability of text $P(\boldsymbol{x})$ and applying it as-is to fill the cloze or prefix prompts defined to specify the task. This is traditionally called the *zero-shot* setting, as there is zero training data for the task of interest.

However, there are also methods that use training data to train the model in concert with prompting methods. These consist of either *full-data learning*, where a reasonably large number of training examples are used to train the model, or *few-shot learning* where a very small number of examples are used to train the model. Prompting methods are particularly useful in the latter case, as there are generally not enough training examples to fully specify the desired behavior, and thus using a prompt to push the model in the right direction is particularly effective.

One thing to note is that for many of the prompt engineering methods described in §4, although annotated training samples are not explicitly used in the training of the downstream task model, they *are* often used in the construction or validation of the prompts that the downstream task will use. As noted by Perez et al. (2021), this is arguably not true zero-shot learning with respect to the downstream task.

7.2 Parameter Update Methods

In prompt-based downstream task learning, there are usually two types of parameters, namely those from (1) pre-trained models and (2) prompts. Which part of parameters should be updated is one important design decision, which can lead to different levels of applicability in different scenarios. We summarize five tuning strategies (as shown in Tab. 6) based on (i) whether the parameters of the underlying LM are tuned, (ii) whether there are additional prompt-related parameters, (iii) if there are additional prompt-related parameters, whether those parameters are tuned.

Strategy	LM Params	Prompt Params		Example
		Additional	Tuned	
Promptless Fine-tuning	Tuned	-		ELMo [130], BERT [32], BART [94]
Tuning-free Prompting	Frozen	✗	✗	GPT-3 [16], AutoPrompt [159], LAMA [133]
Fixed-LM Prompt Tuning	Frozen	✓	Tuned	Prefix-Tuning [96], Prompt-Tuning [91]
Fixed-prompt LM Tuning	Tuned	✗	✗	PET-TC [153], PET-Gen [152], LM-BFF [46]
Prompt+LM Fine-tuning	Tuned	✓	Tuned	PADA [8], P-Tuning [103], PTR [56]

Table 6: Characteristics of different tuning strategies. "Additional" represents if there are additional parameters beyond LM parameters while "Tuned" denotes if parameters are updated.

7.2.1 Promptless Fine-tuning

As mentioned in the introduction, the *pre-train and fine-tune* strategy has been widely used in NLP since before the popularization of prompting methods. Here we refer to pre-training and fine-tuning *without* prompts as *promptless fine-tuning*, to contrast with the prompt-based learning methods introduced in the following sections. In this strategy, given a dataset of a task, all (or some (Howard and Ruder, 2018; Peters et al., 2019)) of the parameters of the pre-trained LM will be updated via gradients induced from downstream training samples. Typical examples of pre-trained models tuned in this way include BERT [32] and RoBERTa [105]. This is a simple, powerful, and widely-used method, but it may overfit or not learn stably on small datasets (Dodge et al., 2020). Models are also prone to *catastrophic forgetting*, where the LM loses its ability to do things that it was able to do before fine-tuning (McCloskey and Cohen, 1989).

- **Advantages**: Simplicity, no need for prompt design. Tuning all the LM parameters allows the model to fit to larger training datasets.
- **Disadvantages**: LMs may overfit or not learn stably on smaller datasets.

7.2.2 Tuning-free Prompting

Tuning-free prompting directly generates the answers without changing the parameters of the pre-trained LMs based only on a prompt, as described in the simplest incarnation of prompting in §2. These can be optionally augmenting input with answered prompts as described in §6.2, and this combination of tuning-free prompting and prompt augmentation is also referred to as *in-context learning* (Brown et al., 2020). Typical examples of tuning-free prompting include LAMA [133] and GPT-3 [16].

- **Advantages**: Efficiency, there is no parameter update process. No catastrophic forgetting, as LM parameters remain fixed. Applicable in zero-shot settings.
- **Disadvantages**: Because prompts are the only method that provide the task specification, heavy engineering is necessary to achieve high accuracy. In particular in the in-context learning setting, providing many answered prompts can be slow at test time, and thus cannot easily use large training datasets.

7.2.3 Fixed-LM Prompt Tuning

In the scenario where additional prompt-relevant parameters are introduced besides parameters of the pre-trained model, *fixed-LM prompt tuning* updates only the prompts' parameters using the supervision signal obtained from the downstream training samples, while keeping the entire pre-trained LM unchanged. Typical examples are Prefix-Tuning [96] and WARP [55].

- **Advantages**: Similarly to tuning-free prompting, it can retain knowledge in LMs and is suitable in few-shot scenarios. Often superior accuracy to tuning-free prompting.
- **Disadvantages**: Not applicable in zero-shot scenarios. While effective in few-shot scenarios, representation power is limited in large-data settings. Prompt engineering through choice of hyperparameters or seed prompts is necessary. Prompts are usually not human-interpretable or manipulable.

7.2.4 Fixed-prompt LM Tuning

Fixed-prompt LM tuning tunes the parameters of the LM, as in the standard pre-train and fine-tune paradigm, but additionally uses prompts with fixed parameters to specify the model behavior. This potentially leads to improvements, particularly in few-shot scenarios.

The most natural way to do so is to provide a discrete textual template that is applied to every training and test example. Typical examples include PET-TC [153], PET-Gen [152], LM-BFF [46]. Logan IV et al. (2021) more recently observe that the prompt engineering can be reduced by allowing for a combination of answer engineering and partial LM fine-tuning. For example, they define a very simple template, *null prompt*, where the input and mask are directly concatenated "`[X][Z]`" without any template words, and find this achieves competitive accuracy.

- **Advantages**: Prompt or answer engineering more completely specify the task, allowing for more efficient learning, particularly in few-shot scenarios.
- **Disadvantages**: Prompt or answer engineering are still required, although perhaps not as much as without prompting. LMs fine-tuned on one downstream task may not be effective on another one.

7.2.5 Prompt+LM Tuning

In this setting, there are prompt-relevant parameters, which can be fine-tuned together with the all or some of the parameters of the pre-trained models. Representative examples include PADA [8], P-Tuning [103]. Notably, this setting is very similar to the standard pre-train and fine-tune paradigm, but the addition of the prompt can provide additional bootstrapping at the start of model training.

- **Advantages**: This is the most expressive method, likely suitable for high-data settings.
- **Disadvantages**: Requires training and storing all parameters of the models. May overfit to small datasets.

8 Applications

In previous sections, we examined prompting methods from the point of view of the mechanism of the method itself. In this section, we rather organize prompting methods from the point of view of which applications they have been applied to. We list these applications in Tab. 7-8 and summarize them in the following sections.

8.1 Knowledge Probing

Factual Probing *Factual probing* (a.k.a. fact retrieval) is one of the earliest scenarios with respect to which prompting methods were applied. The motivation of exploring this task is to quantify how much factual knowledge the pre-trained LM's internal representations bear. In this task, parameters of pre-trained models are usually fixed, and knowledge is retrieved by transforming the original input into a cloze prompt as defined in §2.2, which can be manually crafted or automatically discovered. Relevant datasets including `LAMA` (Petroni et al., 2019) and `X-FACTR` (Jiang et al., 2020a). Since the answers are pre-defined, fact retrieval only focuses on finding effective templates and analyzing the results of different models using these templates. Both discrete template search (Petroni et al., 2019, 2020; Jiang et al., 2020c,a; Haviv et al., 2021; Shin et al., 2020; Perez et al., 2021) and continuous template learning (Qin and Eisner, 2021; Liu et al., 2021b; Zhong et al., 2021b) have been explored within this context, as well as prompt ensemble learning (Jiang et al., 2020c; Qin and Eisner, 2021).

Linguistic Probing Besides factual knowledge, large-scale pre-training also allows LMs to handle linguistic phenomena such as analogies (Brown et al., 2020), negations (Ettinger, 2020), semantic role sensitivity (Ettinger, 2020), semantic similarity (Sun et al., 2021), cant understanding (Sun et al., 2021), and rare word understanding (Schick and Schütze, 2020). The above knowledge can also be elicited by presenting *linguistic probing* tasks in the form of natural language sentences that are to be completed by the LM.

8.2 Classification-based Tasks

Prompt-based learning has been widely explored in classification-based tasks where prompt templates can be constructed relatively easily, such as text classification (Yin et al., 2019) and natural language inference (Schick and Schütze, 2021a). The key to prompting for classification-based tasks is reformulating it as an appropriate prompt. For example, Yin et al. (2019) use a prompt such as "the topic of this document is `[Z]`.", which is then fed into mask pre-trained LMs for slot filling.

Text Classification For *text classification* tasks, most previous work has used cloze prompts, and both prompt engineering (Gao et al., 2021; Hambardzumyan et al., 2021; Lester et al., 2021) and answer engineering (Schick and Schütze, 2021a; Schick et al., 2020; Gao et al., 2021) have been explored extensively. Most existing works explore the efficacy of prompt learning for text classification in the context of *few-shot* setting with "*fixed-prompt LM Tuning*" strategies (defined in §7.2.4).

Natural Language Inference (NLI) NLI aims to predict the relationship (e.g., `entailment`) of two given sentences. Similar to text classification tasks, for *natural language inference* tasks, cloze prompts are commonly used (Schick and Schütze, 2021a). Regarding prompt engineering, researchers mainly focus on the template search in the few-shot learning setting and the answer space $\mathcal{Z}$ is usually manually pre-selected from the vocabulary.

Work	Task	PLM	Setting	Prompt Engineering			Answer Engineering			Tuning	Mul-Pr
				Shape	Man	Auto	Shape	Man	Auto		
LMComm [173]	CR	L2R	Zero	Clo	✓	-	Sp	✓	-	TFP	-
GPT-2 [140]	CR,QA SUM,MT	GPT-2	Zero,Few	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
WNLaMPro [150]	LCP	BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
LMDiagnose [39]	CR,LCP	BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
AdvTrigger [177]	GCG	GPT-2	Full	Pre	-	Disc	Sen	✓	-	TFP	-
CohRank [31]	CKM	BERT	Zero	Clo	✓	-	Tok,Sp	✓	-	TFP	-
LAMA [133]	FP	Conv,Trans ELMo,BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
CTRL [75]	GCG	CTRL	Full	Pre	✓	-	Sen	✓	-	LMT	-
T5 [141]	TC,SUM QA,MT	T5	Full	Pre	✓	-	Tok,Sp,Sen	✓	-	LMT	-
Neg & Mis [74]	FP	Trans,ELMo BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
LPAQA [68]	FP	BERT,ERNIE	Full	Clo	✓	Disc	Tok	✓	-	TFP	PE
ZSC [135]	TC	GPT-2	Full	Pre	✓	-	Tok,Sp	✓	-	LMT	-
PET-TC [153]	TC	RoBERTa,XLM-R	Few	Pre	✓	-	Tok	✓	Disc	LMT	PE
ContxFP [132]	FP	BERT,RoBERTa	Zero	Clo	✓	Disc	Tok	✓	-	TFP	-
UnifiedQA [76]	QA	T5,BART	Full	Prefix	✓	-	Tok,Sp,Sen	✓	-	LMT	-
RAG [95]	QA,GCG,TC	BART	Full	Pre	-	Disc	Tok,Sp,Sen	✓	-	LMPT	PE
GPT-3 [16]	QA,MT,GCG CR,TC,LCP MR,SR,AR	GPT-3	Zero,Few	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
CommS2S [187]	CR	T5	Full	Pre	✓	-	Tok	✓	-	LMT	-
PET-SGLUE [154]	TC	ALBERT	Few	Clo	✓	-	Tok,Sp	✓	-	LMT	PE
ToxicityPrompts [47]	GCG	GPT-1,GPT-2 GPT-3,CTRL	Zero	Pre	✓	-	N/A			TFP	-
WhyLM [147]	Theory	GPT-2	Full	Pre	✓	-	Tok	✓	-	PT	-
X-FACTR [66]	FP	mBERT,BERT XLM,XLM-R	Zero	Clo	✓	-	Tok,Sp	✓	-	TFP	-
Petal [149]	TC	RoBERTa	Few	Clo	✓	-	Tok	-	Disc	LMT	PE
AutoPrompt [159]	TC,FP,IE	BERT,RoBERTa	Full	Clo	-	Disc	Tok	-	Disc	TFP	-
CTRLsum [59]	SUM	BART	Full	Pre	✓	-	Sen	✓	-	LMT	-
PET-Gen [152]	SUM	PEGASUS	Few	Pre	✓	-	Sen	✓	-	LMT	PE
LM-BFF [46]	TC	RoBERTa	Few	Clo	-	Disc	Tok	-	Disc	LMT	PE,PA
WARP [55]	TC	RoBERTa	Few,Full	Clo,Pre	✓	Cont	Tok	✓	Cont	PT	PE
Prefix-Tuning [96]	D2T,SUM	GPT-2,BART	Full	Pre	-	Cont	Sen	✓	-	PT	-
KATE [100]	TC,D2T,QA	GPT-3	Few	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
PromptProg [145]	MT,MR AR,QA	GPT-3	Zero,Few	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
ContxCalibrate [201]	TC,FP,IE	GPT-2,GPT-3	Few	Pre	✓	-	Tok,Sp	✓	-	TFP	PA
PADA [8]	TC,TAG	T5	Full	Pre	-	Disc	N/A			LMPT	-
SD [155]	GCG	GPT-2	Zero	Pre	✓	-	N/A			TFP	-
BERTese [58]	FP	BERT	Full	Clo	✓	Disc	Tok	✓	-	TFP	-
Prompt2Data [148]	TC	RoBERTa	Full	Clo	✓	-	Tok,Sp	✓	-	LMT	-
P-Tuning [103]	FP,TC	GPT-2,BERT ALBERT	Few,Full	Clo,Pre	✓	Cont	Tok,Sp	✓	-	TFP,LMPT	-
GLM [37]	TC	GLM	Full	Clo	✓	-	Tok,Sp	✓	-	LMT	-

Table 7: An organization of works on prompting (Part 1). See the caption of Tab. 8 for a detailed description for all the abbreviations used in this table.

Work	Task	PLM	Setting	Prompt Engineering			Answer Engineering			Tuning	Mul-Pr
				Shape	Man	Auto	Shape	Man	Auto		
ADAPET [170]	TC	ALBERT	Few	Clo	✓	-	Tok,Sp	✓	-	LMT	-
Meta [202]	TC	T5	Full	Pre	✓	-	Tok	✓	-	LMT	-
OptiPrompt [203]	FP	BERT	Full	Clo	✓	Cont	Tok	✓	-	PT	-
Soft [137]	FP	BERT,BART RoBERTa	Full	Clo	✓	Cont	Tok	✓	-	PT	PE
DINO [151]	GCG	GPT-2	Zero	Pre	✓	-	N/A			TFP	-
AdaPrompt [21]	IE	BERT	Few,Full	Clo	✓	-	Tok	-	Disc	LMT	-
PMI_{DC} [62]	GCG,QA,TC	GPT-2,GPT-3	Zero	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	-
Prompt-Tuning [91]	TC	T5	Full	Pre	-	Cont	Tok,Sp	✓	-	PT	PE
Natural-Instr [120]	GCG	GPT-3,BART	Few,Full	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP,LMT	PA
OrderEntropy [111]	TC	GPT-2,GPT-3	Few	Pre	✓	-	Tok	✓	-	TFP	PA
FewshotSemp [158]	SEMP	GPT-3	Few	Pre	✓	-	Sen	✓	-	TFP	PA
PanGu-α [194]	QA,CR,TC SUM,GCG	PanGu-α	Zero,Few	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
TrueFewshot [129]	TC,FP	GPT-2,GPT-3 ALBERT	Few	Clo,Pre	✓	Disc	Tok,Sp	✓	-	TFP,LMT	-
PTR [56]	IE	RoBERTa	Full	Clo	✓	Cont	Tok,Sp	✓	-	LMPT	PC
TemplateNER [29]	TAG	BART	Few,Full	Clo,Pre	✓	-	Tok	✓	-	LMT	PD
PERO [83]	TC,FP	BERT,RoBERTa	Few	Pre	✓	-	Tok	✓	-	TFP	PA
PromptAnalysis [181]	Theory	BERT	Full	Clo	-	Cont	N/A			PT	-
CPM-2 [198]	QA,MR,SUM TC,GCG,MT	CPM-2	Full	Pre	-	Cont	Tok,Sp,Sent	✓	-	PT,LMPT	-
BARTScore [193]	EVALG	BART	Zero	Pre	✓	Disc	Sen	✓	-	TFP	PE
NullPrompt [109]	TC	RoBERTa,ALBERT	Few	Pre	✓	-	Tok	✓	-	LMPT	-
Frozen [174]	VQA,VFP,MG	GPT-like	Full	Pre	-	Cont	Sp (Visual)	✓	-	PT	PA
ERNIE-B3 [167]	TC,LCP,NLI CR,QA,SUM GCG	ERNIE-B3	Zero	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	-
Codex [20]	CodeGen	GPT	Zero,Few Full	Pre	✓	-	Span	✓	Disc	TFP,LMT	PA
HTLM [1]	TC,SUM	BART	Zero,Few Full	Clo	✓	Disc	Tok,Sp,Sen	✓	-	LMT	PA
FLEX [15]	TC	T5	Zero,Few	Pre	✓	-	Tok,Sp	✓	-	LMT	-

Table 8: An organization of works on prompting (Part 2). The **Task** column lists the tasks that are performed in corresponding papers. We use the following abbreviations. **CR**: Commonsense Reasoning. **QA**: Question Answering. **SUM**: Summarization. **MT**: Machine Translation. **LCP**: Linguistic Capacity Probing. **GCG**: General Conditional Generation. **CKM**: Commonsense Knowledge Mining. **FP**: Fact Probing. **TC**: Text Classification. **MR**: Mathematical Reasoning. **SR**: Symbolic Reasoning. **AR**: Analogical Reasoning. **Theory**: Theoretical Analysis. **IE**: Information Extraction. **D2T**: Data-to-text. **TAG**: Sequence Tagging. **SEMP**: Semantic Parsing. **EVALG**: Evaluation of Text Generation. **VQA**: Visual Question Answering. **VFP**: Visual Fact Probing. **MG**: Multimodal Grounding. **CodeGen**: Code generation. The **PLM** column lists all the pre-trained LMs that have been used in corresponding papers for downstream tasks. **GPT-like** is an autoregressive language model which makes small modifications to the original GPT-2 architecture. For other pre-trained LMs, please refer to §3 for more information. **Setting** column lists the settings for prompt-based learning, can be zero-shot learning (**Zero**), few-shot learning (**Few**), fully supervised learning (**Full**). Under **Prompt Engineering**, **Shape** denotes the shape of the template (**Clo** for cloze and **Pre** for prefix), **Man** denotes whether human effort is needed, **Auto** denotes data-driven search methods (**Disc** for discrete search, **Cont** for continuous search). Under **Answer Engineering**, **Shape** indicates the shape of the answer (**Tok** for token-level, **Sp** for span-level, **Sen** for sentence- or document-level), and **Man** and **Auto** are the same as above. The **Tuning** column lists tuning strategies (§7). **TFP**: Tuning-free Prompting. **LMT**: Fixed-prompt LM Tuning. **PT**: Fixed-LM Prompt Tuning. **LMPT**: LM+Prompt Tuning. The **Mul-Pr** column lists multi-prompt learning methods. **PA**: Prompt Augmentation. **PE**: Prompt Ensembling. **PC**: Prompt Composition. **PD**: Prompt Decomposition.

8.3 Information Extraction

Unlike classification tasks where cloze questions can often be intuitively constructed, for *information extraction* tasks constructing prompts often requires more finesse.

Relation Extraction *Relation extraction* is a task of predicting the relation between two entities in a sentence. Chen et al. (2021b) first explored the application of *fixed-prompt LM Tuning* in relation extraction and discuss two major challenges that hinder the direct inheritance of prompting methodology from classification tasks: (1) The larger label space (e.g. 80 in relation extraction v.s 2 in binary sentiment classification) results in more difficulty in answer engineering. (2) In relation extraction, different tokens in the input sentence may be more or less important (e.g. entity mentions are more likely to participate in a relation), which, however, can not be easily reflected in the prompt templates for classification since the original prompt template regards each word equally. To address the above problems, Chen et al. (2021b) propose an adaptive answer selection method to address the issue (1) and task-oriented prompt template construction for the issue (2), where they use special markers (e.g. `[E]`) to highlight the entity mentions in the template. Similarly, Han et al. (2021) incorporate entity type information via multiple prompt composition techniques (illustrated in Fig. 4).

Semantic Parsing *Semantic parsing* is a task of generating a structured meaning representation given a natural language input. Shin et al. (2021) explore the task of few-shot semantic parsing using LMs by (1) framing the semantic parsing task as a paraphrasing task (Berant and Liang, 2014) and (2) constraining the decoding process by only allowing output valid according to a grammar. They experiment with the *in-context learning* setting described in §7.2.2, choosing answered prompts that are semantically close to a given test example (determined by the conditional generation probability of generating a test sample given another training example). The results demonstrate the effectiveness of the paraphrasing reformulation for semantic parsing tasks using pre-trained LMs.

Named Entity Recognition *Named entity recognition* (NER) is a task of identifying named entities (e.g., person name, location) in a given sentence. The difficulty of prompt-based learning's application to tagging tasks, exemplified as NER, is that, unlike classification, (1) each unit to be predicted is a token or span instead of the whole input text, (2) there is a latent relationship between the token labels in the sample context. Overall, the application of prompt-based learning in tagging task has not been fully explored. Cui et al. (2021) recently propose a template-based NER model using BART, which enumerates text spans and considers the generation probability of each type within manually crafted templates. For example, given an input "Mike went to New York yesterday", to determine what type of entity "Mike" is, they use the template "Mike is a `[Z]` entity", and the answer space $\mathcal{Z}$ consists of values such as "person" or "organization".

8.4 "Reasoning" in NLP

There is still a debate[6] about if deep neural networks are capable of performing "reasoning" or just memorizing patterns based on large training data (Arpit et al., 2017; Niven and Kao, 2019). As such, there have been a number of attempts to probe models' reasoning ability by defining benchmark tasks that span different scenarios. We detail below how prompting methods have been used in these tasks.

Commonsense Reasoning There are a number of benchmark datasets testing commonsense reasoning in NLP systems (Huang et al., 2019; Rajani et al., 2019; Lin et al., 2020; Ponti et al., 2020). Some commonly attempted tasks involve solving Winograd Schemas (Levesque et al., 2012), which require the model to identify the antecedent of an ambiguous pronoun within context, or involve completing a sentence given multiple choices. For the former, an example could be "The trophy doesn't fit into the brown suitcase because it is too large." And the task for the model is to infer whether "it" refers to the trophy or the "suitcase". By replacing "it" with its potential candidates in the original sentences and calculating the probability of the different choices, pre-trained LMs can perform quite well by choosing the choice that achieves the highest probability (Trinh and Le, 2018). For the latter, an example could be "Eleanor offered to fix her visitor some coffee. Then she realized she didn't have a clean `[Z]`.". The candidate choices are "cup", "bowl" and "spoon". The task for the pre-trained LM is to choose the one from the three candidates that most conforms to common sense. For these kinds of tasks, we can also score the generation probability of each candidate and choose the one with the highest probability (Ettinger, 2020).

Mathematical Reasoning Mathematical reasoning is the ability to solve mathematical problems, e.g. arithmetic addition, function evaluation. Within the context of pre-trained LMs, researchers have found that pre-trained embeddings and LMs can perform simple operations such as addition and subtraction when the number of digits is small, but fail when the numbers are larger (Naik et al., 2019; Wallace et al., 2019b; Brown et al., 2020). Reynolds and McDonell (2021) explore more complex mathematical (e.g. $f(\boldsymbol{x}) = \boldsymbol{x} * \boldsymbol{x}$, what is $f(f(3))$?) reasoning problems and improve LM performance through serializing reasoning for the question.

[6] e.g. https://medium.com/reconstruct-inc/the-golden-age-of-computer-vision-338da3e471d1

8.5 Question Answering

Question answering (QA) aims to answer a given input question, often based on a context document. QA can take a variety of formats, such as extractive QA (which identifies content from the context document containing the answer; e.g. SQuAD (Rajpurkar et al., 2016)), multiple-choice QA (where the model has to pick from several choices; e.g. RACE (Lai et al., 2017)), and free-form QA (where the model can return an arbitrary textual string as a response; e.g. NarrativeQA (Kočiský et al., 2018)). Generally, these different formats have been handled using different modeling frameworks. One benefit of solving QA problems with LMs, potentially using prompting methods, is that different formats of QA tasks can be solved within the same framework. For example, Khashabi et al. (2020) reformulate many QA tasks as a text generation problem by fine-tuning seq2seq-based pre-trained models (e.g. T5) and appropriate prompts from the context and questions. Jiang et al. (2020b) take a closer look at such prompt-based QA systems using sequence to sequence pre-trained models (T5, BART, GPT2) and observe that probabilities from these pre-trained models on QA tasks are not very predictive of whether the model is correct or not.

8.6 Text Generation

Text generation is a family of tasks that involve generating text, usually conditioned on some other piece of information. Prompting methods can be easily applied to these tasks by using *prefix prompts* together with autoregressive pre-trained LMs. Radford et al. (2019) demonstrated impressive ability of such models to perform generation tasks such as text summarization and machine translation using prompts such as "translate to french, `[X]`, `[Z]`". Brown et al. (2020) perform *in-context learning* (§7.2.2) for text generation, creating a prompt with manual templates and augmenting the input with multiple *answered prompts*. Schick and Schütze (2020) explore *fixed-prompt LM tuning* (§7.2.4) for few-shot text summarization with manually crafted templates. (Li and Liang, 2021) investigate *fixed-LM prompt tuning* (§7.2.3) for text summarization and data-to-text generation in few-shot settings, where learnable prefix tokens are prepended to the input while parameters in pre-trained models are kept frozen. Dou et al. (2021) explored the *prompt+LM tuning* strategy (§7.2.5) on text summarization task, where learnable prefix prompts are used and initialized by different types of guidance signals, which can then be updated together with parameters of pre-trained LMs.

8.7 Automatic Evaluation of Text Generation

Yuan et al. (2021b) have demonstrated that prompt learning can be used for automated evaluation of generated texts. Specifically, they conceptualize the evaluation of generated text as a text generation problem, modeled using a pre-trained sequence-to-sequence, and then use *prefix prompts* that bring the evaluation task closer to the pre-training task. They experimentally find that simply adding the phrase "such as" to the translated text when using pre-trained models can lead to a significant improvement in correlation on German-English machine translation (MT) evaluation.

8.8 Multi-modal Learning

Tsimpoukelli et al. (2021) shift the application of prompt learning from text-based NLP to the *multi-modal* setting (vision and language). Generally, they adopt the *fixed-LM prompt tuning* strategy together with *prompt augmentation* techniques. They specifically represent each image as a sequence of continuous embeddings, and a pre-trained LM whose parameters are frozen is prompted with this prefix to generate texts such as image captions. Empirical results show few-shot learning ability: with the help of a few demonstrations (answered prompts), system can rapidly learn words for new objects and novel visual categories.

8.9 Meta-Applications

There are also a number of applications of prompting techniques that are not NLP tasks in and of themselves, but are useful elements of training strong models for any application.

Domain Adaptation Domain adaptation is the practice of adapting a model from one domain (e.g. news text) to another (e.g. social media text). Ben-David et al. (2021) use self-generated *domain related features* (DRFs) to augment the original text input and perform sequence tagging as a sequence-to-sequence problem using a seq2seq pre-trained model.

Debiasing Schick et al. (2021) found that LMs can perform self-diagnosis and self-debiasing based on biased or debiased instructions. For example, to self-diagnosis whether the generated text contains violent information, we can use the following template "The following text contains violence. `[X][Z]`". Then we fill `[X]` with the input text and look at the generation probability at `[Z]`, if the probability of "Yes" is greater than "No", then we would assume the given text contains violence, and vice versa. To perform debiasing when generating text, we first compute the probability of the next word $P(x_t|\boldsymbol{x}_{<t};\theta)$ given the original input. Then we compute the probability

of next word $P(x_t|[\boldsymbol{x}_{<t}; \boldsymbol{x}_{\text{diagnosis}}]; \theta)$ by appending self-diagnosis textual input to the original input as mentioned above. These two probability distributions for the next token can be combined to suppress the undesired attribute.

Dataset Construction Schick and Schütze (2021) propose to use pre-trained LMs to generate datasets given certain instructions. As an example, suppose we have an unlabeled dataset in which each sample is a sentence. If we want to construct a dataset containing pairs of semantically similar sentences, then we can use the following template for each input sentence: "Write two sentences that mean the same thing. `[X] [Z]`" and attempt to generate a sentence that shares the same meaning as the input sentence.

8.10 Resources

We also collect some useful resources for different prompt-based applications.

Dataset Some datasets specifically designed for few-shot and zero-shot learning are shown in Tab. 9.

Task	Dataset	Setting	URL
Commonsense Reasoning	Pronoun Disambiguation Problems [93]	Zero	https://cs.nyu.edu/ davise/papers/...
	Winograd Schema Challenge [93]	Zero	https://cs.nyu.edu/ davise/papers/...
	CPRAG-102 [39]	Zero	https://github.com/aetting/lm-diagnostics
Linguistic Capacity Probing	WNLaMPro [150]	Zero	https://github.com/timoschick/...
	ROLE-88 [39]	Zero	https://github.com/aetting/lm-diagnostics
	NEG-136 [39]	Zero	https://github.com/aetting/lm-diagnostics
Fact Probing	LAMA [133]	Zero	https://dl.fbaipublicfiles.com/LAMA/...
	Negated LAMA [74]	Zero	https://github.com/norakassner/LAMA...
	Misprimed LAMA [74]	Zero	https://github.com/norakassner/LAMA...
	X-FACTR [66]	Zero	https://x-factr.github.io/
	LAMA-TREx-easy-hard [203]	Zero	https://github.com/princeton-nlp/...
Text Classification	FLEX [15]	Zero,Few	https://github.com/allenai/flex
	FewGLUE [154]	Few	https://github.com/timoschick/fewglue
General Conditional Gen.	REALTOXICITYPROMPTS [47]	Zero	https://allenai.org/data/...
	Natural-Instructions [120]	Few,Full	https://instructions.apps.allenai.org/

Table 9: Few-shot and zero-shot datasets for prompt-based learning.

Prompts As shown in Tab. 10, we collect existing commonly-used prompts designed manually, which can be regarded as off-the-shelf resource for future research and applications.

9 Prompt-relevant Topics

What is the essence of prompt-based learning and how does it relate to other learning methods? In this section, we connect prompt learning with other similar learning methods.

Ensemble Learning *Ensemble learning* (Ting and Witten, 1997; Zhou et al., 2002) is a technique that aims to improve the performance of a task by taking advantage of the complementarity of multiple systems. Generally, the different systems used in an ensemble result from different choices of architectures, training strategies, data ordering, and/or random initialization. In prompt ensembling (§6.1), the choice of prompt templates becomes another way to generate multiple results to be combined. This has the clear advantage that this does not necessarily require training the model multiple times. For example, when using discrete prompts, these prompts can simply be changed during the inference stage (Jiang et al., 2020c).

Few-shot Learning *Few-shot learning* aims to learn a machine learning system in the data-scarce scenarios with few training samples. There are a wide variety of methods to achieve few-shot learning including model agnostic meta-learning (Finn et al., 2017b) (learning features rapidly adaptable to new tasks), embedding learning (Bertinetto et al., 2016) (embedding each sample in a lower-dimensional space where similar samples are close together), memory-based learning (Kaiser et al., 2017) (representing each sample by a weighted average of contents from the memory) etc. (Wang et al., 2020). Prompt augmentation can be regarded as another way to achieve few-shot learning (a.k.a. priming-based few-shot learning (Kumar and Talukdar, 2021)). Compared to previous methods, prompt augmentation directly prepends several labeled samples to the currently-processed sample elicit knowledge from pre-trained LMs even without any parameter tuning.

Task	Example Prompt-Answer	Resource
Fact Probing	**Prompt** Adolphe Adam died in `[Z]`. **Answer** $\mathcal{V}$ **Prompt** iPod Touch is produced by `[Z]`. **Answer** $\mathcal{V}$ **Prompt** The official language of Mauritius is `[Z]`. **Answer** $\mathcal{V}$	LAMA dataset LPAQA dataset X-FACTR dataset
Text Classificatin	**Prompt** Which of these choices best describes the following document? "`[Class A]`", "`[Class B]`", "`[Class C]`". `[X][Z]` **Answer** `[Class A]`, `[Class B]`, `[Class C]` **Prompt** How is the text best described? : "`[Class A]`", "`[Class B]`" , or "`[Class C]`". `[X][Z]` **Answer** `[Class A]`, `[Class B]`, `[Class C]` **Prompt** This passage is about `[Z]`: `[X]` **Answer** `[Class A]`, `[Class B]`, `[Class C]` **Prompt** `[X]`. Is this review positive? `[Z]` **Answer** Yes, No **Prompt** `[X]` It was `[Z]`. **Answer** great, terrible	Meta [202]
Natural Language Inference	**Prompt** `[X1]`? `[Z]`, `[X2]` **Answer** Yes, No, Maybe **Prompt** `[X1]` `[Z]`, `[X2]` **Answer** Yes, No, Maybe	
Commonsense Reasoning	**Prompt** The trophy doesn't fit into the brown suitcase because `[Z]` is too large. **Answer** trophy, suitcase **Prompt** Ann asked Mary what time the library closes, because `[Z]` had forgotten. **Answer** Ann, Mary	PDP dataset WSC dataset CPRAG-102 dataset
Linguistic Knowledge Probing	**Prompt** A robin is a `[Z]`. **Answer** bird, tree **Prompt** A robin is not a `[Z]`. **Answer** bird, tree **Prompt** New is the opposite of `[Z]`. **Answer** old, young, current	WNLaMPro dataset ROLE-88 dataset NEG-136 dataset
Named Entity Recognition	**Prompt-Pos** `[X]` `[Span]` is a `[Z]` entity. **Prompt-Neg** `[X]` `[Span]` is not a named entity. **Answer** person, location, organization, miscellaneous **Prompt-Pos** The entity type of `Span` is `[Z]`. **Prompt-Neg** `[X]` The entity type of `[Span]` is none entity. Answer person, location, organization, miscellaneous	TemplateNER [29]
Question Answering	**Prompt** `[Question]` `[Passage]` `[Z]` **Prompt** `[Passage]` According to the passage, `[Question]` `[Z]` **Prompt** Based on the following passage, `[Question]` `[Z]`. `[Passage]`	
Summarization	**Prompt** Text: `[X]` Summary: `[Z]` **Prompt** `[X]` TL;DR: `[Z]` **Prompt** `[X]` In summary, `[Z]`	BARTScore [193]
Machine Translation	**Prompt** French: `[French sentence]` English: **Prompt** A French sentence is provided: `[French sentence]` The French translator translates the sentence into English: `[Z]` **Prompt** `[French sentence]` = `[Z]`	

Table 10: Commonly used prompts and answers for different tasks. `[X]` and `[Z]` denote slots for input and answer respectively. $\mathcal{V}$ denotes the vocabulary of the LM. More prompts for each task can be found using the **Resource** column.

Prompt Concept	Relevant Topic	Commonality	Peculiarity
Prompt Ensembling [68; 153]	Ensemble Learning [171; 204]	Combine results of multiple systems to get better performance	In prompt ensembling, multiple predictions result from different prompt variants. This contrasts with architecture or feature variations, each of which requires separate training.
Prompt Augmentation [16; 46]	Few-shot Learning [160; 42]	Use few examples to learn generalized rules	Prompt augmentation is a specific subset of few-shot learning.
	Larger-context Learning [18; 53]	Introduce larger context to aid the learning process	Additional information introduced in larger-context learning is not necessarily the labeled data.
Discrete Prompt Search [68; 159]	Query reformulation [123; 123]	Reformulate the input into a query form	Query reformulation commonly focuses on information extraction and question answering tasks, while prompt learning can be applied to a variety of NLP tasks
Discrete Prompt Fine-tuning [46]	QA-based multi-task learning [115; 97]	Reformulate many tasks into an QA form	QA-based formulations aim to solve different tasks through question answering, while prompting additionally targets full use of pre-trained models.
Continuous Prompt Fine-tuning [103; 36]	Controlled Text Generation [191; 77; 156]	Input is augmented with additional inputs to control the generation process	Controlled generation targets generation of a particular type of text while prompt learning uses prompts to specify the task itself.
Prompt-based downstream task learning [153; 193]	Supervised Attention [101; 165]	Require external hint to remind the model of which part information should be focused on	Research works on supervised attention usually target at salient information from an image or text, while prompt learning aims to utilize relevant knowledge from the pre-trained model.
	Data augmentation [40; 144]	Improving downstream tasks' performance by introducing additional samples	Data augmentation introduce additional training samples in an explicit way while prompts can be regarded as highly-condensed training samples [88].

Table 11: Other research topics relevant to prompting methods.

Larger-context Learning *Larger-context learning* aims to improve the system's performance by augmenting the input with additional contextual information, e.g. retrieved from the training set (Cao et al., 2018) or external data sources (Guu et al., 2020). Prompt augmentation can be regarded as adding relevant labeled samples into the input, but a minor difference is in larger-context learning, the introduced context is not necessarily labeled data.

Query Reformulation *Query reformulation* (Mathieu and Sabatier, 1986; Daumé III and Brill, 2004) is commonly used in information retrieval (Nogueira and Cho, 2017) and question answering tasks (Buck et al., 2017; Vakulenko et al., 2020), which aim to elicit more relevant texts (documents or answers) by expanding the input query with related query terms (Hassan, 2013) or generating paraphrases. There are several commonalities between prompt-based learning and query reformulation, for example (1) both aim to make better use of some existing knowledge bases by asking a right questions (2) the knowledge bases are usually a black-box, not available to the users, so researchers must learn how to probe it optimally based on solely questions.

There are also differences: the knowledge base in traditional query reformulation problems is usually a search engine (Nogueira and Cho, 2017), or QA system (Buck et al., 2017). By contrast, for prompt-based learning, we usually define this knowledge base as an LM, and need to find the appropriate query to elicit an appropriate answer from it. The input reformulation in prompt learning has changed the form of tasks. For example, an original text classification task has been converted into a cloze question problem, therefore bringing additional complexity regarding how to (1) make an appropriate task formulation, and (2) change the modeling framework accordingly. These steps are not required in traditional query formulation. Despite these discrepancies, some methodologies from query reformulation research still can be borrowed for prompt learning, such as decomposing input query into multiple sub-queries (Nogueira et al., 2019), similar to prompt decomposition.

QA-based Task Formulation *QA-based task formulation* aims to conceptualize different NLP tasks as a question-answering problem. (Kumar et al., 2016; McCann et al., 2018) are earlier works that attempt to unify multiple NLP tasks into a QA framework. Later, this idea has been further explored in information extraction (Li et al., 2020; Wu

et al., 2020) and text classification (Chai et al., 2020). These methods are very similar to the prompting methods introduced here in that they use textual questions to specify which task is to be performed. However, one of the key points of prompting methods is how to better use the knowledge in pre-trained LMs, and these were not covered extensively on previous works advocating for QA formulations.

Controlled Generation *Controlled generation* aims to incorporate various types of guidance beyond the input text into the generation model (Yu et al., 2020). Specifically, the guidance signals could be *style tokens* (Sennrich et al., 2016b; Fan et al., 2018), *length specifications* (Kikuchi et al., 2016), *domain tags* (Chu et al., 2017), or any variety of other pieces of information used to control of the generated text. It could also be *keywords* (Saito et al., 2020), *relation triples* (Zhu et al., 2020) or even *highlighted phrases or sentences* (Grangier and Auli, 2018; Liu et al., 2021c) to plan the content of generated texts. In a way, many of the prompting methods described here are a type of controllable generation, where the prompt is usually used to specify the *task itself*. Thus, it is relatively easy to find commonalities between the two genres: (1) both add extra information to the input text for better generation, and these additional signals are (often) learnable parameters. (2) If "controlled generation" is equipped with seq2seq-based pre-trained models (e.g., BART), then it is can be regarded as prompt learning with input-dependent prompts and the *prompt+LM fine-tuning* strategy (§7.2.5), e.g. *GSum* (Dou et al., 2021), where both the prompt's and pre-trained LM's parameters can be tuned.

Also, some clear discrepancies between controlled generation and prompt-based text generation are: (1) In controlled generation work, the control is generally performed over the style or content of the generations (Fan et al., 2018; Dou et al., 2021) while the underlying task remains the same. They don't necessarily require a pre-trained model. In contrast, the main motivation for using prompts for text generation is to specify the task itself and better utilize the pre-trained model. (2) Moreover, most of the current work on prompt learning in text generation shares a dataset- or task-level prompt (Li and Liang, 2021). Only very few works have explored input-dependent ones (Tsimpoukelli et al., 2021). However, this is a common setting and effective in the controlled text generation, which may provide valuable direction for the future work on prompt learning.

Supervised Attention Knowing to pay attention to the important information is a key step when extracting useful information from objects such as long text sequences (Liu et al., 2016; Sood et al., 2020), images (Sugano and Bulling, 2016; Zhang et al., 2020b), or knowledge bases (Yu et al., 2020; Dou et al., 2021)). *Supervised attention* (Liu et al., 2017b) aims to provide explicit supervision over the attention of models based on the fact that completely data-driven attention can overfit to some artifacts (Liu et al., 2017a). In this respect, prompt learning and supervised attention share ideas that both aim to extract salient information with some clues, which need to be provided separately. To solve this problem, supervised attention methods tried to use additional loss functions to learn to predict gold attention on a manually labeled corpus (Jiang et al., 2015; Qiao et al., 2018; Gan et al., 2017). Research on prompt learning may also borrow ideas from this literature.

Data Augmentation Data augmentation is a technique that targets increasing the amount of data that can be used for training by making modifications to existing data (Fadaee et al., 2017; Ratner et al., 2017). As recently observed by (Scao and Rush, 2021), adding prompts can achieve a similar accuracy improvement to the addition of 100s of data points on average across classification tasks, which suggests that using prompts for a downstream task is similar to conducting data augmentation implicitly.

10 Challenges

Although prompt-based learning has shown significant potential among different tasks and scenarios, several challenges remain, some of which we detail below.

10.1 Prompt Design

Tasks beyond Classification and Generation Most existing works about prompt-based learning revolve around either text classification or generation-based tasks. Applications to information extraction and text analysis tasks have been discussed less, largely because the design of prompts is less straightforward. We expect that applying prompting methods to these tasks in the future it will require either reformulating these tasks so that they can be solved using classification or text generation-based methods, or performing effective answer engineering that expresses structured outputs in an appropriate textual format.

Prompting with Structured Information In many NLP tasks, the inputs are imbued with some variety of structure, such as tree, graph, table, or relational structures. How to best express these structures in prompt or answer engineering is a major challenge. Existing works (Chen et al., 2021b) make a step by making prompts with additional marks to encode lexical information, such as entity markings. Aghajanyan et al. (2021) present structured prompts based on hyper text markup language for more fine-grained web text generation. However, moving beyond this to more complicated varieties of structure is largely unexplored, and a potentially interesting research area.

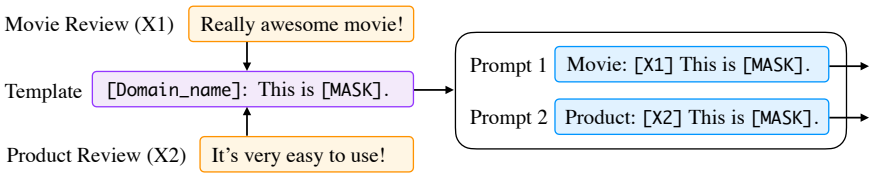


Figure 5: Multi-prompt learning for multi-task, multi-domain or multi-lingual learning. We use different colors to differentiate different components as follows. " " for input text, " " for template, " " for prompt.

Entanglement of Template and Answer The performance of a model will depend on *both* the templates being used and the answer being considered. How to simultaneously search or learn for the best combination of template and answer remains a challenging question. Current works typically select answers before select template (Gao et al., 2021; Shin et al., 2020), but Hambardzumyan et al. (2021) have demonstrated the initial potential of simultaneously learning both.

10.2 Answer Engineering

Many-class and Long-answer Classification Tasks For classification-based tasks, there are two main challenges for answer engineering: (a) When there are too many classes, how to select an appropriate answer space becomes a difficult combinatorial optimization problem. (b) When using multi-token answers, how to best decode multiple tokens using LMs remains unknown, although some multi-token decoding methods have been proposed (Jiang et al., 2020a).

Multiple Answers for Generation Tasks For text generation tasks, qualified answers can be semantically equivalent but syntactically diverse. So far, almost all works use prompt learning for text generation relying solely on a single answer, with only a few exceptions (Jiang et al., 2020c). How to better guide the learning process with multiple references remains a largely open research problem.

10.3 Selection of Tuning Strategy

As discussed in §7, there are a fairly wide variety of methods for tuning parameters of prompts, LMs, or both. However, given the nascent stage of this research field, we still lack a systematic understanding of the tradeoffs between these methods. The field could benefit from systematic explorations such as those performed in the pre-train and fine-tune paradigm regarding the tradeoffs between these different strategies (Peters et al., 2019).

10.4 Multiple Prompt Learning

Prompt Ensembling In prompt ensembling methods, the space and time complexity increase as we consider more prompts. How to distill the knowledge from different prompts remains underexplored. Schick and Schütze (2020, 2021a,b) use an ensemble model to annotate a large dataset to distill the knowledge from multiple prompts.

In addition, how to select ensemble-worthy prompts is also under-explored. For text generation tasks, the study of prompt ensemble learning has not been performed so far, probably because ensemble learning in text generation itself is relatively complicated. To remedy this problem, some recently proposed neural ensembling methods such as *Refactor* (Liu et al., 2021c) could be considered as a method for prompt ensembling in text generation tasks.

Prompt Composition and Decomposition Both prompt composition and decomposition aim to break down the difficulty of a complicated task input by introducing multiple sub-prompts. In practice, how to make a good choice between them is a crucial step. Empirically, for those token (Ma and Hovy, 2016) or span (Fu et al., 2021) prediction tasks (e.g., NER), prompt decomposition can be considered, while for those span relation prediction (Lee et al., 2017) tasks (e.g., entity coreference), prompts composition would be a better choice. In the future, the general idea of de-/composing can be explored in more scenarios.

Prompt Augmentation Existing prompt augmentation methods are limited by the input length, i.e., feeding too many demonstrations to input is infeasible. Therefore, how to select informative demonstrations, and order them in an appropriate is an interesting but challenging problem (Kumar and Talukdar, 2021).

Prompt Sharing All the above considerations refer to the application of prompt in a single task, domain or language. We may also consider *prompt sharing*, where prompt learning is applied to multiple tasks, domains, or languages. Some key issues that may arise include how to design individual prompts for different tasks, and how to modulate their interaction with each other. So far this field has not been explored. Fig.5 illustrates a simple multiple prompt learning strategy for multiple tasks, where prompt templates are partially shared.

10.5 Selection of Pre-trained Models

With plenty of pre-trained LMs to select from (see §3), how to choose them to better leverage prompt-based learning is an interesting and difficult problem. Although we have conceptually introduced (§3.4) how different paradigms of pre-trained models are selected for diverse NLP tasks, there are few to no systematic comparisons of the benefits brought by prompt-based learning for different pre-trained LMs.

10.6 Theoretical and Empirical Analysis of Prompting

Despite their success in many scenarios, theoretical analysis and guarantees for prompt-based learning are scarce. Wei et al. (2021) showed that soft-prompt tuning can relax the non-degeneracy assumptions (the generation probability of each token is linearly independent) needed for downstream recovery (i.e. recover the ground-truth labels of the downstream task.), making it easier to extract task-specific information. Saunshi et al. (2021) verified that text classification tasks can be reformulated as sentence completion tasks, thus making language modeling a meaningful pre-training task. Scao and Rush (2021) empirically show that prompting is often worth 100s of data points on average across classification tasks.

10.7 Transferability of Prompts

Understanding the extent to which prompts are specific to the model and improving the transferability of prompts are also important topics. (Perez et al., 2021) show that prompts selected under tuned few-shot learning scenario (where one has a larger validation set to choose prompts) generalize well across models of similar sizes while prompts selected under true few-shot learning scenario (where one only has a few training samples) do not generalize as effectively as the former setting among models with similar sizes. The transferability is poor when the model sizes are quite different in both scenarios.

10.8 Combination of Different Paradigms

Notably, much of the success of the prompting paradigm is built on top of pre-trained models that were developed for the pre-train and fine-tune paradigm, such as BERT. However, are the pre-training methods that are effective for the latter applicable as-is to the former, or can we entirely re-think our pre-training methods to further improve accuracy or ease of applicability to prompting-based learning? This is an important research question that has not been covered extensively by the literature.

10.9 Calibration of Prompting Methods

Calibration (Gleser, 1996) refers to the ability of a model to make good probabilistic predictions. When using the generation probability of the pre-trained LMs (e.g., BART) to predict the answer, we need to be careful since the probability distribution is typically not well calibrated. Jiang et al. (2020b) observed the probabilities of pre-trained models (e.g., BART, T5, GPT-2) on QA tasks are well calibrated. Zhao et al. (2021) identify three pitfalls (majority label bias, recency bias and common token bias) that lead the pre-trained LMs to be biased toward certain answers when provided answered prompts. For example, if the final answered prompt has a positive label, then this will bias the model towards predicting positive words. To overcome those pitfalls, Zhao et al. (2021) first use context-free input (e.g. the prompt would be "Input: Subpar acting. Sentiment: Negative\n Input: Beautiful film. Sentiment: Positive\n Input: N/A. Sentiment:") to get the initial probability distribution P_0, then they use the real input (e.g. the prompt would be "Input: Subpar acting. Sentiment: Negative\n Input: Beautiful film. Sentiment: Positive\n Input: Amazing. Sentiment:") to get the probability distribution P_1. Finally, these two distributions can be used to get a calibrated generation probability distribution. However, this method has two drawbacks: (1) it comes with the overhead of finding proper context-free input (e.g. whether to use "N/A" or "None") and (2) the probability distribution of the underlying pre-trained LM is still not calibrated.

Even though we have a calibrated probability distribution, we also need to be careful when we assume a single gold answer for an input. This is because that all surface forms of a same object will compete for finite probability mass (Holtzman et al., 2021). For example, if we consider the gold answer to be "Whirlpool bath", the generation probability of it will typically be low since the word "Bathtub" shares the same meaning and it will take over a large probability mass. To address this issue, we could either (i) perform answer engineering to construct a comprehensive gold answer set using paraphrasing methods (§5.2.2) or (ii) calibrate the probability of a word based on its prior likelihood within the context (Holtzman et al., 2021).

11 Meta Analysis

In this section, we aim to give a quantitative birds-eye view of existing research on prompting methods by performing a meta analysis over existing research works along different dimensions.

TABLE 12 Timeline of prompt-based learning. The time for each paper is based on its first arXiv version (if exists) or estimated submission time. A web-version can refer to NLPedia-Pretrain. Works in red consider natural language understanding (NLU) tasks; works in blue consider natural language generation (NLG) tasks; works in green consider both NLU tasks and NLG tasks.

2018.06.07	LMComm (Trinh and Le, 2018)
2019.02.14	GPT-2 (Radford et al., 2019)
2019.04.14	WNLaMPro (Schick and Schütze, 2020)
2019.07.31	LMDiagnose (Ettinger, 2020)
2019.08.20	AdvTrigger (Wallace et al., 2019a)
2019.09.02	CohRank (Davison et al., 2019)
2019.09.03	LAMA (Petroni et al., 2019)
2019.09.11	CTRL (Keskar et al., 2019)
2019.10.23	T5 (Raffel et al., 2020)
2019.11.08	Neg & Misprim (Kassner and Schütze, 2020)
2019.11.28	LPAQA (Jiang et al., 2020c)
2019.12.10	ZSC (Puri and Catanzaro, 2019)
2020.01.21	PET-TC (Schick and Schütze, 2021a)
2020.03.10	ContxFP (Petroni et al., 2020)
2020.05.02	UnifiedQA (Khashabi et al., 2020)
2020.05.22	RAG (Lewis et al., 2020b)
2020.05.28	GPT-3 (Brown et al., 2020)
2020.09.08	CommS2S(Yang et al., 2020)
2020.09.15	PET-SGLUE (Schick and Schütze, 2021b)
2020.09.24	ToxicityPrompts (Gehman et al., 2020)
2020.10.07	WhyLM (Saunshi et al., 2021)
2020.10.13	X-FACTR (Jiang et al., 2020a)
2020.10.26	Petal (Schick et al., 2020)
2020.10.29	AutoPrompt (Shin et al., 2020)
2020.12.08	CTRLsum (He et al., 2020a)
2020.12.22	PET-Gen (Schick and Schütze, 2020)
2020.12.31	LM-BFF (Gao et al., 2021)
2021.01.01	WARP (Hambardzumyan et al., 2021)
2021.01.01	Prefix-Tuning (Li and Liang, 2021)
2021.01.17	KATE (Liu et al., 2021a)
2021.02.15	PromptProg (Reynolds and McDonell, 2021)
2021.02.19	ContxCalibrate (Zhao et al., 2021)
2021.02.24	PADA (Ben-David et al., 2021)
2021.02.27	SD (Schick et al., 2021)
2021.03.09	BERTese (Haviv et al., 2021)
2021.03.15	Prompt2Data (Scao and Rush, 2021)
2021.03.18	P-Tuning (Liu et al., 2021b)
2021.03.18	GLM (Du et al., 2021)
2021.03.22	ADAPET (Tam et al., 2021)
2021.04.10	Meta (Zhong et al., 2021a)
2021.04.12	OptiPrompt (Zhong et al., 2021b)
2021.04.14	Soft (Qin and Eisner, 2021)
2021.04.15	DINO (Schick and Schütze, 2021)
2021.04.15	AdaPrompt (Chen et al., 2021b)
2021.04.16	PMI_{DC} (Holtzman et al., 2021)
2021.04.18	Prompt-Tuning (Lester et al., 2021)
2021.04.18	Natural-Instr (Mishra et al., 2021)
2021.04.18	OrderEntropy (Lu et al., 2021)
2021.04.18	FewshotSemp (Shin et al., 2021)
2021.04.26	PanGu-α (Zeng et al., 2021)
2021.05.24	TrueFewshot (Perez et al., 2021)
2021.05.24	PTR (Han et al., 2021)
2021.06.03	TemplateNER (Cui et al., 2021)
2021.06.03	PERO (Kumar and Talukdar, 2021)
2021.06.16	PromptAnalysis (Wei et al., 2021)
2021.06.20	CPM-2 (Zhang et al., 2021)
2021.06.21	BARTScore (Yuan et al., 2021b)
2021.06.24	NullPrompt (Logan IV et al., 2021)
2021.06.25	Frozen (Tsimpoukelli et al., 2021)
2021.07.05	ERNIE-B3 (Sun et al., 2021)
2021.07.07	Codex (Chen et al., 2021a)
2021.07.14	HTLM (Aghajanyan et al., 2021)
2021.07.15	FLEX (Bragg et al., 2021)

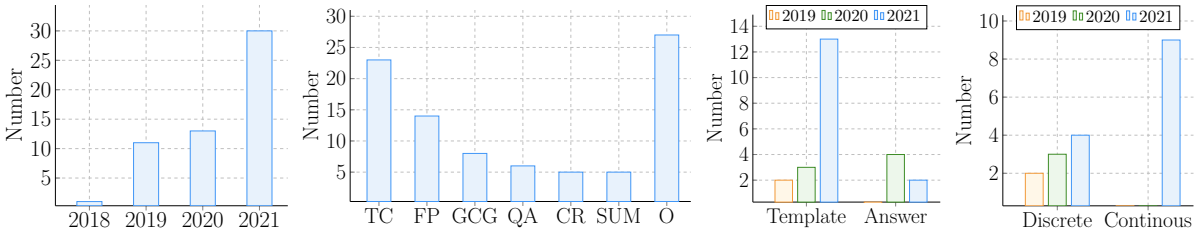
11.1 Timeline

We first summarize a number of existing research papers in a chronological order with in the form of a *timeline*, which hopefully, help researchers who are new to this topic understand the evolution of the field.

11.2 Trend Analysis

We also calculate the number of prompt-based papers with respect to different dimensions.

Year With the emergence of different kinds of pre-trained LMs, prompt-based learning has become a more and more active research field, as can be seen in Fig. 6-(a). We can see a huge surge in 2021, which is perhaps due to the prevalence of GPT-3 (Brown et al., 2020), which greatly increased the popularity of prompting in the few-shot multi-task setting.



(a) Year. (b) Task. (c) Automatic Search. (d) Search Space.

Figure 6: Meta-analyses over different dimensions. The statistics are based on the works in Tab. 7 and Tab. 8. In (d), we use the following abbreviations. TC: text classification, FP: factual probing, GCG: general conditional generation, QA: question answering, CR: commonsense reasoning, SUM: summarization, O: others.

Tasks We plot the number of works that investigate various tasks in Fig. 6-(b). For a task that has fewer than 5 relevant works, we group it into "Others". As the bar chart indicates, most tasks regarding prompt-based learning revolve around text classification and factual probing. We conjecture that this is because that for these tasks, both template engineering and answer engineering are relatively easy to conduct, and experiments are relatively computationally inexpensive.

Prompt vs. Answer Search As noted in previous sections, both prompt and answer search are important tools to take advantage of pre-trained language models for many tasks. Current research mainly focuses on template search instead of answer search, as shown in Fig. 6-(c).

Likely reasons are: (1) For conditional generation tasks (e.g. summarization or translation), the gold references can be directly used as answer. Although there are many sequences that may share the same semantics, how to effectively conduct multi-reference learning in conditional text generation problems is non-trivial. (2) For classification tasks, most of the time, label words are relative easy to select using domain knowledge.

Discrete Search vs. Continuous Search Since there are only a few works focus on automatic answer search, we analyze the automatic template search. As time goes by, there has been a shift from discrete search to continuous search for prompt engineering, as shown in Fig. 6-(d). Likely reasons are: (1) discrete search is harder to optimize compared to continuous search, (2) soft prompts have greater representation ability.

12 Conclusion

In this paper, we have summarized and analyzed several paradigms in the development of statistical natural language processing techniques, and have argued that *prompt-based learning* is a promising new paradigm that may represent another major change in the way we look at NLP. First and foremost, we hope this survey will help researchers more effectively and comprehensively understand the paradigm of prompt-based learning, and grasp its core challenges so that more scientifically meaningful advances can be made in this field. In addition, looking all the way back to the summary of the four paradigms of NLP research presented in §1, we hope to highlight the commonalities and differences between them, making research on any of these paradigms more full-fledged, and potentially providing a catalyst to inspire work towards the next paradigm shift as well.

Acknowledgements

We would like to thank Chunting Zhou for her constructive comments on this work.