Augmented Language Models: a Survey

Anonymous authors
Paper under double-blind review

Abstract

This survey reviews works in which language models (LMs) are augmented with reasoning skills and the ability to use tools. The former is defined as decomposing a potentially complex task into simpler subtasks while the latter consists in calling external modules such as a code interpreter. LMs can leverage these augmentations separately or in combination via heuristics, or learn to do so from demonstrations. While adhering to a standard missing tokens prediction objective, such augmented LMs can use various, possibly non-parametric external modules to expand their context processing ability, thus departing from the pure language modeling paradigm. We therefore refer to them as Augmented Language Models (ALMs). The missing token objective allows ALMs to learn to reason, use tools, and even act, while still performing standard natural language tasks and even outperforming most regular LMs on several benchmarks. In this work, after reviewing current advance in ALMs, we conclude that this new research direction has the potential to address common limitations of traditional LMs such as interpretability, consistency, and scalability issues.

1 Introduction: motivation for the survey and definitions

1.1 Motivation and Definitions

Large Language Models (LLMs) (Devlin et al., 2019; Brown et al., 2020; Chowdhery et al., 2022) have fueled dramatic progress in Natural Language Processing (NLP) and are already core in several products with millions of users, such as the coding assistant Copilot (Chen et al., 2021), Google search engine[1] or more recently ChatGPT[2]. Memorization (Tirumala et al., 2022) combined with compositionality (Zhou et al., 2022) capabilities made LLMs able to execute various tasks such as language understanding or conditional and unconditional text generation at an unprecedented level of performance, thus opening a realistic path towards higher-bandwidth human-computer interactions.

However, LLMs suffer from important limitations hindering a broader deployment. LLMs often provide non-factual but seemingly plausible predictions, often referred to as hallucinations (Welleck et al., 2020). This leads to many avoidable mistakes, for example in the context of arithmetics (Qian et al., 2022) or within a reasoning chain (Wei et al., 2022c). Moreover, many LLMs groundbreaking capabilities seem to emerge with size, measured by the number of trainable parameters: for example, Wei et al. (2022b) demonstrate that LLMs become able to perform some BIG-bench tasks[3] via few-shot prompting once a certain scale is attained. Although a recent line of work yielded smaller LMs that retain some capabilities from their largest counterpart (Hoffmann et al., 2022), the size and need for data of LLMs can be impractical for training but also maintenance: continual learning for large models remains an open research question (Scialom et al., 2022). Other limitations of LLMs are discussed by Goldberg (2023) in the context of *ChatGPT*, a chatbot built upon *GPT3*.

We argue these issues stem from a fundamental defect of LLMs: they are generally trained to perform statistical language modeling given (i) a single parametric model and (ii) a limited context, typically the n

[1] See *e.g.* https://blog.google/products/search/search-language-understanding-bert/
[2] https://openai.com/blog/chatgpt/
[3] https://github.com/google/BIG-bench

previous or surrounding tokens. While n has been growing in recent years thanks to software and hardware innovations, most models still use a relatively small context size compared to the potentially large context needed to always correctly perform language modeling. Hence, massive scale is required to store knowledge that is not present in the context but necessary to perform the task at hand.

As a consequence, a growing research trend emerged with the goal to solve these issues, slightly moving away from the pure statistical language modeling paradigm described above. For example, a line of work circumvents the limited context size of LLMs by increasing its relevance: this is done by adding information extracted from relevant external documents. Through equipping LMs with a module that retrieves such documents from a database given a context, it is possible to match certain capabilities of some of the largest LMs while having less parameters (Borgeaud et al., 2022; Izacard et al., 2022). Note that the resulting model is now non-parametric, since it can query external data sources. More generally, LMs can also improve their context via reasoning strategies (Wei et al. (2022c); Taylor et al. (2022); Yang et al. (2022c) *inter alia*) so that a more relevant context is produced in exchange for more computation before generating an answer. Another strategy is to allow LMs to leverage external tools (Press et al. (2022); Gao et al. (2022); Liu et al. (2022b) *inter alia*) to augment the current context with important missing information that was not contained in the LM's weights. Although most of these works aim to alleviate the downfalls of LMs mentioned above separately, it is straightforward to think that more systematically augmenting LMs with both reasoning and tools may lead to significantly more powerful agents. We will refer to these models as **Augmented Language Models (ALMs)**. As this trend is accelerating, keeping track and understanding the scope of the numerous results becomes arduous. This calls for a taxonomy of ALMs works and definitions of technical terms that are used with sometimes different intents.

Definitions. We now provide definitions for terms that will be used throughout the survey.

- **Reasoning.** In the context of ALMs, reasoning is decomposing a potentially complex task into simpler subtasks the LM can solve more easily by itself or using tools. There exist various ways to decompose into subtasks, such as recursion or iteration. In that sense, reasoning is akin to planning as defined for example in LeCun (2022). In this survey, reasoning will very often refer to the various strategies to improve reasoning skills in LMs, such as step-by-step reasoning using few-shot examples. It is not yet fully understood whether the LM is really reasoning, or simply producing a larger context that increases the likelihood of correctly predicting the missing tokens. We refer to Huang and Chang (2022) for a discussion on this topic: although reasoning may currently be an abuse of language given the current state of the art, the term is already in use within the community. A more pragmatic definition of reasoning in the context of ALMs is giving more computation steps to the model before yielding the answer to a prompt.

- **Tool.** For ALMs, a tool is an external module that is typically called using a rule or a special token and whose output is included in the ALM's context. The tool can gather external information, or have an effect on the virtual or physical world (generally perceived by the ALM). An example of a tool fetching external information is a document retriever, while a tool having an external effect could be a robotic arm. A tool can be called at training or at inference time. More generally, learning to interact with a tool may consist in learning to call its API.

- **Act.** For ALMs, calling a tool that modifies a state in a virtual or physical object, and observing the result, typically by including it in the ALM's current context. For example, some works from the survey discuss searching the web, or robotic arm manipulation via LMs. With a slight abuse of term, we will sometimes denote the call of a tool by an ALM as an action, even if it does not have an external effect.

Why jointly discussing reasoning and tools? The combination of reasoning and tools within LMs should allow solving a broad range of complex tasks without heuristics, hence with better generalization capabilities. Typically, reasoning would foster the LM to decompose a given problem into potentially simpler subtasks while tools would help getting each step right, for example obtaining the result from a mathematical operation. Put it differently, reasoning is a way for LMs to combine different tools in order to solve complex

tasks, and tools are a way to not fail a reasoning with valid decomposition. Both should benefit from the other. Moreover, reasoning and tools can be put under the same hood, as both augment the context of the LM so that it better predicts the missing tokens, albeit in a different way.

Why jointly discussing tools and actions? Tools that gather additional information and tools that have an effect on the virtual or physical world can be called in the same fashion by the LM. For example, there is seemingly no difference between a LM outputting python code for solving a mathematical operation, and a LM outputting python code to manipulate a robotic arm. A few works discussed in the survey are already using LMs that have effects on the virtual or physical world: under this view, we can say that the LM have the potential to act, and expect important advances in the direction of LMs as autonomous agents.

1.2 Our classification

We decompose the works included in the survey under three axes. Section 2 studies works which augment LM's reasoning capabilities as defined above. Section 3 focuses on works allowing LMs to interact with external tools and act. Finally, Section 4 explores whether reasoning and tools usage are implemented via heuristics or learned, *e.g.* via supervision or reinforcement. Other axes could naturally have been chosen for this survey and are discussed in Section 5. For conciseness, the survey focuses on works that combine reasoning or tools with LMs. However, the reader should keep in mind that many of these techniques were originally introduced in another context than LMs, and consult the introduction and related work section of the papers we mention if needed. Finally, although we focus on LLMs, not all works we consider employ large models, hence we stick to LMs for correctness in the remainder of the survey.

2 Reasoning

In general, reasoning is the ability to make inferences using evidence and logic. Reasoning can be divided into multiple types of skills such as commonsense reasoning (McCarthy et al., 1960; Levesque et al., 2012), mathematical reasoning (Cobbe et al., 2021), symbolic reasoning (Wei et al., 2022c), etc. Often, reasoning involves deductions from inference chains, called as multi-step reasoning. In the context of LMs, we will use the definition of reasoning provided in Section 1. Previous work has shown that LLMs can solve simple reasoning problems but fail at complex reasoning (Creswell et al., 2022): hence, this section focuses on various strategies to augment LM's reasoning skills. One of the challenges with complex reasoning problems for LMs is to correctly obtain the solution by composing the correct answers predicted by it to the sub-problems. For example, a LM may correctly predict the dates of birth and death of a celebrity, but may not correctly predict the age. Press et al. (2022) call this discrepancy the compositionality gap for LMs. For the rest of this section, we discuss the works related to three popular paradigms for eliciting reasoning in LMs. Note that Huang and Chang (2022) propose a survey on reasoning in language models. Qiao et al. (2022) also propose a survey on reasoning albeit with a focus on prompting. Since our present work focuses on reasoning combined with tools, we refer the reader to Huang and Chang (2022); Qiao et al. (2022) for a more in-depth review of works on reasoning for LLMs.

2.1 Eliciting reasoning with prompting

In recent years, prompting LMs to solve various downstream tasks has become a dominant paradigm (Brown et al., 2020). In prompting, examples from a downstream task are transformed such that they are formulated as a language modeling problem. Prompting typically takes one of the two forms: zero-shot, where the model is directly prompted with a test example's input; and few-shot, where few examples of a task are prepended along with a test example's input. This few-shot prompting is also known as in-context learning or few-shot learning. As opposed to "naive" prompting that requires an input to be directly followed by the output/answer, elicitive prompts encourage LMs to solve tasks by following intermediate steps before predicting the output/answer. While Nye et al. (2021) provides the first example of few-shot prompting LLMs with reasoning examples and Cobbe et al. (2021) generalizes the use of reasoning examples to non-algorithmic tasks, Wei et al. (2022c) extensively studies how elicitive prompting enables LMs to be better reasoners in a

Question: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
Answer: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.
Question: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Answer:
<LM>

Figure 1: An example of few-shot Chain-of-Thought prompt. **<LM>** denotes call to the LM with the above prompt.

few-shot setting. Later, Kojima et al. (2022) showed similar ability in a zero-shot setting. We discuss them in detail in the following paragraphs.

Few-shot setting. Wei et al. (2022c) popularized chain-of-thought (CoT), a few-shot prompting technique for LMs. The prompt consists of examples of a task, with inputs followed by intermediate reasoning steps leading to the final output, as depicted in Figure 1. Table 1 shows that CoT outperforms standard prompting methods. Wei et al. (2022b) observe that the success of the few-shot strategy emerges with scale, while Tay et al. (2022) add that without fine-tuning, successful use of CoT generally requires 100B+ parameters LMs such as *LaMDA* (Thoppilan et al., 2022), *PaLM* (Chowdhery et al., 2022) or *GPT3* (Brown et al., 2020; Ouyang et al., 2022), before proposing *UL2*, a 20B open source model that can perform CoT. Using few-shot CoT prompting, *Minerva* (Lewkowycz et al., 2022) achieves excellent performance on math benchmarks such as GSM8K (Cobbe et al., 2021). Wang et al. (2022c) further improve CoT with *Self-consistency*: diverse reasoning paths are sampled from a given language model using CoT, and the most consistent answer is selected as the final answer. Press et al. (2022) introduce *Self-ask*, a prompt in the spirit of CoT. Instead of providing the model with a continuous chain of thought as in Figure 1, *Self-ask* explicitly states the follow-up question before answering it and relies on a scaffold (e.g, *"Follow-up question:"* or *"So the final answer is:"*), so that the answers are more easily parseable. The authors demonstrate an improvement over CoT on their introduced datasets aiming at measuring the compositionality gap. They observe that this gap does not narrow when increasing the size of the model. Note that Press et al. (2022) focus on 2-hop questions, *i.e.*, questions for which the model only needs to compose two facts to obtain the answer. Interestingly, *Self-ask* can easily be augmented with a search engine (see Section 3). *ReAct* (Yao et al., 2022b) is another few-shot prompting approach eliciting reasoning that can query three tools throughout the reasoning steps: `search` and `lookup` in Wikipedia, and `finish` to return the answer. *ReAct* will be discussed in more detail in the next sections.

Zero-shot setting. Kojima et al. (2022) extend the idea of eliciting reasoning in LMs to zero-shot prompting. Whereas few-shot provides examples of the task at hand, zero-shot conditions the LM on a single prompt that is not an example. Here, Kojima et al. (2022) simply append *Let's think step by step* to the input question before querying the model (see Figure 2), and demonstrate that zero-shot-CoT for large LMs does well on reasoning tasks such as GSM8K although not as much as few-shot-CoT.

Question: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Answer: Let's think step by step
<LM>

Figure 2: An example of zero-shot Chain-of-Thought prompt. **<LM>** denotes call to the LM with the above prompt.

Model	Accuracy (%)
OpenAI (`text-davinci-002`)[1]	15.6
OpenAI (`text-davinci-002`) + CoT[1]	46.9
OpenAI (`text-davinci-002`) + CoT + Calculator[1]	46.9
OpenAI (`code-davinci-002`)[1]	19.7
OpenAI (`code-davinci-002`) + CoT[1]	63.1
OpenAI (`code-davinci-002`) + CoT + Calculator[1]	65.4
GPT-3 175B + FT + CoT + Calculator[2]	34.0
GPT-3 175B + FT + CoT + Calculator + Verifier[2]	55.0
PaLM 540B[3]	17.0
PaLM 540B+CoT[3]	54.0
PaLM 540B+CoT+Calculator[3]	58.0
PAL[4]	72.0

Table 1: Evaluation of different reasoning methods on GSM8K, a popular reasoning benchmark. FT denotes fine-tuning and CoT denotes chain-of-thought. The reported accuracies are based on [1]: (Wei et al., 2022c); [2]: (Cobbe et al., 2021); [3]: (Chowdhery et al., 2022); and [4]: (Gao et al., 2022).

2.2 Recursive prompting

Several works attempt to improve LM's reasoning by explicitly decomposing problems into sub-problems in order to solve the problem in a divide and conquer manner. This recursive approach slightly differs from CoT since the latter does not explicitly formulate sub-problems, and can be especially useful for complex tasks, given that compositional generalization can be challenging for LMs (Lake and Baroni, 2018; Keysers et al., 2019; Li et al., 2022a). Methods that employ problem decomposition can either then solve the sub-problems independently, where these answers are aggregated to generate the final answer (Perez et al., 2020; Min et al., 2019), or solve the sub-problems sequentially, where the solution to the next sub-problem depends on the answer to the previous ones (Yang et al., 2022a; Zhou et al., 2022; Drozdov et al., 2022; Dua et al., 2022; Khot et al., 2022; Wang et al., 2022a; Wu et al., 2022b; Mishra and Nouri, 2022). For instance, in the context of math problems, *Least-to-most* prompting (Zhou et al., 2022) allows a language model to solve harder problems than the demonstration examples by decomposing a complex problem into a list of sub-problems. It first employs few-shot prompting to decompose the complex problem into sub-problems, before sequentially solving the extracted sub-problems, using the solution to the previous sub-problems to answer the next one. Patel et al. (2022) show that modifying existing benchmarks by letting human decompose questions into subquestions that are relatively easier for models to solve leads to significant improvement for *GPT3* and *RoBERTa-SQuAD* equipped with a symbolic calculator.

While many earlier works include learning to decompose through distant supervision (Perez et al., 2020; Talmor and Berant, 2018; Min et al., 2019), like Zhou et al. (2022), many recent works employ in-context learning to do so (Yang et al., 2022a; Khot et al., 2022; Dua et al., 2022; Kazemi et al., 2022). Among these, there are further differences. For instance, Drozdov et al. (2022) is a follow-up work to Zhou et al. (2022), but differs by using a series of prompts to perform recursive syntactic parses of the input rather than a linear decomposition, and also differs by choosing the exemplars automatically through various heuristics. Dua et al. (2022) is concurrent work with Zhou et al. (2022) but differs by interweaving the question decomposition and answering stages, i.e., the next sub-question prediction has access to the previous questions and answers as opposed to generating all sub-questions independently of any previous answers. Yang et al. (2022a), on the other hand, decomposes using rule-based principles and slot-filling prompting to translate questions into a series of SQL operations. Khot et al. (2022) also employs prompts to decompose into specific operations, but then allows each sub-problem to be solved using a library of specialized handlers, where each is devoted to a particular sub-task (e.g., retrieval). Finally, Kazemi et al. (2022) decompose a given problem in a backward fashion: starting from the goal and a set of rules, the system decompose the goal into sub-goals and recursively check whether the sub-goals can be proved. Here again, the modules are implemented by few-shot prompting a pre-trained LM.

Prompt 0

Question: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?
<LM>
Answer: To solve "How many times can she slide before it closes?", we need to first solve: "How long does each trip take?"
</LM>

Prompt 1

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes.
Subquestion 1: How long does each trip take?
<LM>
Answer 1: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.
</LM>

Prompt 2

It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.
Subquestion 1: How long does each trip take?
Answer 1: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.
Subquestion 2: How many times can she slide before it closes?
<LM>
Answer 2: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide 15 ÷ 5 = 3 times before it closes.
</LM>

Figure 3: Recursive prompting example. **<LM>** denotes the start of the LM's output to the prompt, while **</LM>** denotes the end. The problem is first decomposed into subproblems in **Prompt 0**. Then, **Answer 2** to **Subquestion 2** and **Answer 1** to **Subquestion 1** are sequentially fed to **Prompt 2** and **Prompt 1**. The few-shot examples for each stage's prompt are omitted. Inspired from Figure 1 in Zhou et al. (2022).

2.3 Explicitly teaching language models to reason

Despite their spectacular results, prompting approaches have some drawbacks in addition to requiring model scale. Namely, they require to discover prompts that elicit e.g. step-by-step reasoning, manually providing examples when it comes to few-shot for a new task. Moreover, prompting is computationally expensive in the case of long prompts, and it is harder to benefit from a relatively large number of examples due to limited context size of the model. Recent works suggest to circumvent these issues by training LMs to use, as humans, a working memory when more than one step are required to solve a task correctly. Nye et al. (2021) introduce the notion of scratchpad, allowing a LM to better perform on multi-step computation tasks such as addition or code execution. More precisely, at training time, the LM sees input tasks such as addition along with associated intermediate steps: the ensemble is called a scratchpad. At test time, the model is required to predict the steps and the answer from the input task. Scratchpads differ from the above prompting strategies in that they are fine-tuned on example tasks with associated computation steps. Note however that Nye

et al. (2021) also perform experiments in the few-shot regime. Taylor et al. (2022) use a similar approach in the context of large LM pre-training: *Galactica* was trained on a corpus of scientific data including some documents where step-by-step reasoning is wrapped with a special token `<work>` and `</work>` to mimic an internal working memory. At inference time, the model can be asked explicitly to activate this reasoning mode via the `<work>` token. Taylor et al. (2022) argue that one more problem arise when training on reasoning examples: many intermediate reasoning steps may be missing in the training data curated from the internet, as humans do not explicitly write all their reasoning steps. To circumvent the issue of missing steps, the authors created datasets with detailed reasoning process. An example of prompt seen during *Galactica*'s pre-training is presented in Figure 4.

Other recent works improve the reasoning abilities of pre-trained LMs via fine-tuning. Zelikman et al. (2022) propose a bootstrap approach to generate reasoning steps (also called rationales) for a large set of unlabeled data and use that data to fine-tune the model. Lengerich et al. (2022) propose a self-supervised method which extracts reasoning capabilities of a teacher model by asking the question "why" given the student's initial responses to various NLP problems. Next, the student model is fine-tuned by using contrastive distillation via sampling evidence from memory. Yu et al. (2022) show that standard LM fine-tuning on reasoning tasks lead to better reasoning skills such as textual entailment, abductive reasoning, and analogical reasoning, compared to pre-trained models. Further, several instruction fine-tuning approaches (Ouyang et al., 2022; Chung et al., 2022; Iyer et al., 2022; Ho et al., 2022) use chain-of-thought style prompts to achieve remarkable improvements on popular benchmarks such as BBH (Srivastava et al., 2022) and MMLU (Hendrycks et al., 2021). Interestingly, all these works also show that small scale instruction-finetuned models can perform better than un-finetuned large scale models, especially in the tasks where instruction following is important.

Question: A needle 35 mm long rests on a water surface at 20 ∘ C. What force over and above the needle's weight is required to lift the needle from contact with the water surface? $\sigma = 0.0728m$.
`<work>`

$$\begin{aligned} \sigma &= 0.0728N/m \\ \sigma &= F/L \\ 0.0728 &= F/(2 \times 0.035) \\ F &= 0.0728(2 \times 0.035) \end{aligned}$$

```
calculate.py
"'
f = 0.0728*(2*0.035)
with open("output.txt", "w") as file:
file.write(str(round(f, 5)))
"'
```

«run: `calculate.py`»

«read: `output.txt`»

0.0051

`</work>`

Answer: $F = 0.0051N$

Figure 4: Working memory example from Taylor et al. (2022). This prompt and its output are seen during LM pre-training.

2.4 Comparison and limitations of abstract reasoning

Overall, reasoning can be seen as decomposing a problem into a sequence of sub-problems either iteratively or recursively.[4] Exploring as many reasoning paths as possible is hard and there is no guarantee that the intermediate steps are valid. A way to produce faithful reasoning traces is to generate pairs of questions and their corresponding answers for each reasoning step (Creswell and Shanahan, 2022), but there is still no guarantee of the correctness of these intermediate steps. Overall, a reasoning LM seeks to improve its context by itself so that it has more chance to output the correct answer. To what extent LMs actually use the stated reasoning steps to support the final prediction remains poorly understood (Yu et al., 2022).

In many cases, some reasoning steps may suffer from avoidable mistakes that compromise the correctness of the output. For example, mistakes on nontrivial mathematical operations in a reasoning step may lead to the wrong final output. The same goes with known facts such as the identity of a president at a given year. Some of the works studied above (Yao et al., 2022b; Press et al., 2022) already leverage simple external tools such as a `search engine` or a `calculator` to validate intermediate steps. More generally, the next section of the survey focuses on the various tools that can be queried by LMs to increase the chance of outputting a correct answer.

3 Using Tools and Act

A recent line of LM research allows the model to access knowledge that is not necessarily stored in its weights, such as a given piece of factual knowledge. More precisely, tasks such as exact computation or information retrieval for example can be offloaded to external modules such as a `python interpreter` or a `search engine` that are queried by the model which, in that respect, use tools. Additionally, we can say the LM performs an action when the tool has an effect on the external world. The possibility to easily include tools and actions in the form of special tokens is a convenient feature of language modeling coupled with transformers.

3.1 Calling another model

In many cases, the tool can simply be another neural network or the LM itself.

Iterative LM calling. As an alternative to improving the LM's context for better outputs after a single inference pass, an alternative and intuitive way to get better results from LMs consists of repeatedly calling the model to iteratively refine its output. *Re3* (Yang et al., 2022c) exploits this idea to automatically generate stories of over two thousand words. More precisely, *Re3* first generates a plan, setting, and characters by prompting *GPT3* (Brown et al., 2020) with a premise. Then, *Re3* iteratively injects information from both the plan and current story state into a new *GPT3* prompt to generate new story passages. This work is improved upon in Yang et al. (2022b) with the use of a learned detailed outliner that iteratively expands the brief initial outline to any desired level of granularity. Other approaches that teach models to iteratively improve texts in an unsupervised fashion range from applications such as blank filling (Shen et al., 2020; Donahue et al., 2020) to denoising a sequence of Gaussian vectors into word vectors (Li et al., 2022c). *PEER* (Schick et al., 2022), for example, is a model initialized from *LM-Adapted T5* (Raffel et al., 2020) and trained on Wikipedia edits, learning both how to carry out edits and how to plan for the next steps. Consequently, *PEER* is able to develop articles by repeatedly planning and editing as in Figure 5. The iterative approach has the additional benefit of allowing a complex task like story and article generation to be decomposed into smaller subtasks. Importantly and apart from *PEER*, the works mentioned above employ heuristics to call the LM. A future research direction may consist in allowing the LM to call itself repeatedly until the output satisfies a certain criterion. Rather than just calling a single model repeatedly, Wu et al. (2022a) propose an interactive interface for a pipeline allowing chaining of multiple LMs together, where the

[4]Here, reasoning is described as a sequential operation. However, other reasoning structures such as trees could be considered. For example, Lample et al. (2022) leverage trees to model the different strategies leading to a proof for a given theorem. A strategy is a set of intermediate results that must be either true or themselves proved, hence decomposed into another new subset of intermediate results.

Iteration 0

Text: Brittney Reese (born September 9, 1986 in Gulfport, Mississippi) is an American long jumper.
<LM>
Plan: Remove incorrect information
Edit: Brittney Reese (born September 9, 1986 ~~in Gulfport, Mississippi~~) is an American long jumper.
</LM>

Iteration 1

Text: Brittney Reese (born September 9, 1986) is an American long jumper.
<LM>
Plan: Add information about her career
Edit: Brittney Reese (born September 9, 1986) is an American long jumper , who competed at the 2008 Summer Olympics, and is a 4-time World Champion .
</LM>

Iteration 2

Text: Brittney Reese (born September 9, 1986) is an American long jumper, who competed at the 2008 Summer Olympics, and is a 4-time World Champion.
<LM>
Plan: Add her birthplace
Edit: Brittney Reese (born September 9, 1986 in Inglewood, California) is an American long jumper, who competed at the 2008 Summer Olympics, and is a 4-time World Champion.
</LM>

Figure 5: Iterative prompting example using PEER (Schick et al., 2022), a LM trained to produce a plan of action and edit to the input text at each step. This process can be repeated until the generated text requires no further updates. **<LM>** denotes the start of the LM's output to the prompt, while **</LM>** denotes the end.

output of one step is passed as input to the next. Such contributions allow non-AI-experts to refine solutions to complex tasks that cannot be appropriately handled by a single LM.

Leveraging other modalities. Prompts under the form of text may not contain enough context to correctly perform a given task. For example, a question does not call for the same answer if it is asked with a serious or ironic tone. Including various modalities into the context would probably be useful for LMs such as chatbots. As recently demonstrated by Hao et al. (2022) and Alayrac et al. (2022), LMs can also be used as a general-purpose interface with models pre-trained on different modalities. For example, Hao et al. (2022) take a number of pre-trained encoders that can process diverse modalities such as vision and language, and connect them to a LM that serves as a universal task layer. The interface and modular encoders are jointly pre-trained via a semi-causal language modeling objective. This approach combines the benefits of causal and non-causal language modeling, enabling both in-context learning and open-ended generation, as well as easy fine-tuning of the encoders. Similarly, Alayrac et al. (2022) introduce *Flamingo*, a family of Visual Language Models (VLMs) that can handle any interleaved sequences of visual and textual data. *Flamingo* models are trained on large-scale multimodal web corpora containing interleaved text and images, which enables them to display in-context few-shot learning capabilities of multimodal tasks. With only a handful of annotated examples, *Flamingo* can easily adapt to both generation tasks such as visual question-answering

and captioning, as well as classification tasks such as multiple-choice visual question-answering. Zeng et al. (2022) introduce Socratic Models, a modular framework in which various models pre-trained on different modalities can be composed zero-shot. This allows models to exchange information with each other and acquire new multimodal capabilities without additional finetuning. Socratic Models enable new applications such as robot perception and planning, free-form question-answering about egocentric videos, or multimodal assistive dialogue by interfacing with external APIs and databases such as search engines. Interestingly, other modalities such as images can be incorporated to improve reasoning capabilities of moderate size LMs (1B) (Zhang et al., 2023), and enabling multimodal chain of thought reasoning (Lu et al., 2022a).

3.2 Information retrieval

LMs can be augmented with memory units, for example via a neural cache of recent inputs (Grave et al., 2017; Merity et al., 2017), to improve their reasoning abilities. Alternatively, knowledge in the form of natural language can be offloaded completely from the LM by retrieving from an external knowledge source. Memory augmentation strategies help the language model to avoid producing non-factual and out-of-date information as well as reducing the number of parameters required to achieve comparable performance to large LMs.

3.2.1 Retrieval-augmented language models

Dense and sparse retrievers. There exist two types of retrievers that can be used to augment a LM: dense and sparse. Sparse retrievers work with sparse bag-of-words representations of the documents and the queries (Robertson and Zaragoza, 2009). In contrast, dense neural retrievers use a dense query and dense document vectors obtained from a neural network (Asai et al., 2021). Both types of retrievers assess the relevance of a document to an information-seeking query. This can be done by (i) checking for precise term overlap or (ii) computing the semantic similarity across related concepts. Sparse retrievers excel at the first sub-problem, while dense retrievers can be better at the second (Luan et al., 2021).

Conditioning LMs on retrieved documents. Various works augment LMs with a `dense retriever` by adding the retrieved documents to the current context (Chen et al., 2017; Clark and Gardner, 2017; Lee et al., 2019; Guu et al., 2020; Khandelwal et al., 2020; Lewis et al., 2020; Izacard and Grave, 2020; Zhong et al., 2022; Borgeaud et al., 2022; Izacard et al., 2022; Shi et al., 2023). Even though the idea of retrieving documents to perform question answering is not new, retrieval-augmented LMs have recently demonstrated strong performance in other knowledge intensive tasks besides Q&A. These proposals close the performance gap compared to larger LMs that use significantly more parameters. *REALM* (Guu et al., 2020) was the first method to jointly train end-to-end a retrieval system with an encoder LM. *RAG* (Lewis et al., 2020) jointly fine-tunes the retriever with a sequence-to-sequence model. Izacard and Grave (2020) introduced a modification of the seq2seq architecture to efficiently process many retrieved documents. Borgeaud et al. (2022) focuses on an auto-regressive LM, called *RETRO*, and use a large-scale corpus indexed by frozen *BERT* embeddings for the retriever module. Crucially, the authors show that at scale (retrieval corpus and language model), this approach does not require to train and update the retriever. In particular, *RETRO* obtains comparable performance to *GPT3* on different downstream tasks. Although *RETRO* was trained with retrieval from scratch, their approach allows the integration of retrieval into existing pre-trained LMs. *Atlas* (Izacard et al., 2022) jointly trains a retriever with a sequence-to-sequence model to obtain a LM with strong few-shot learning capabilities in spite of being orders of magnitude smaller than many other large LMs. Table 2 compares the main characteristics of the models discussed, notably how the retrieval results are integrated into the LM's context. In all the aforementioned cases, the query corresponds to the prompt but this has been relaxed, see the chain-of-thought subsection below.

Efficient large scale retrieval. Since retrieval-augmented language models store knowledge in an external data store, it is crucial that the information retrieval step is efficient, especially when dealing with a large number of document/passage/sentence embeddings and/or a large number (billions) of query vectors. The literature offers a variety of optimisations to boost performance of retrievers as well as reducing the memory footprint of the data store.

Model	# Retrieval tokens	Granularity	Retriever training	Retrieval integration
REALM (Guu et al., 2020)	$O(10^9)$	Prompt	End-to-End	Append to prompt
RAG (Lewis et al., 2020)	$O(10^9)$	Prompt	Fine-tuning	Cross-attention
RETRO (Borgeaud et al., 2022)	$O(10^{12})$	Chunk	Frozen	Chunked cross-attn.
Atlas (Izacard et al., 2022)	$O(10^9)$	Prompt	Fine-tuning	Cross-attention

Table 2: Comparison between database retrieval augmented languages models. Inspired by Table 3 from Borgeaud et al. (2022).

First, an index structure can be built for the document/passage/sentence embeddings to perform efficient similarity search, for instance, using the Facebook AI Similarity Search library (faiss) (Johnson et al., 2021). Leveraging indexing structures enables efficient search operations by partitioning the vector space and enabling fast pruning of irrelevant vectors.

Second, if there exist memory constraints due to a large number of document/passage/sentence embeddings, a multi-node, multi-gpu distributed framework can be used to store the index corresponding to only part of the document/passage/sentence embeddings per process. The query embeddings can also be split in batches and obtain the top-k results in parallel using gpus for further speed ups for approximate or exact search. When exact search becomes intractable due to scale, approximate nearest neighbor algorithms can be used. These algorithms trade off accuracy for speed, allowing significantly faster retrieval while maintaining reasonably accurate results, see Johnson et al. (2021) for further details.

Finally, in order to further reduce the memory footprint of each of the index shards corresponding to a subset of the document/passage/sentence embeddings, different compression techniques can be used. For instance, the Atlas retrieval-augmented language model (Izacard et al., 2022) uses product quantisation (Jégou et al., 2011) to reduce the memory footprint of the retriever without sacrificing accuracy in downstream tasks in terms of exact match and recall@50 metrics of Q&A.

Currently, there also exist a variety of vector database frameworks that have some of these optimisations implemented out of the box and can be leveraged to use retrieval as a service without requiring the user to implement all optimisations from scratch.

Chain-of-thought prompting and retrievers. Recent works (He et al., 2022; Trivedi et al., 2022) propose to combine a `retriever` with reasoning via chain-of-thoughts (CoT) prompting to augment a LM. He et al. (2022) use the CoT prompt to generate reasoning paths consisting of an explanation and prediction pair. Then, knowledge is retrieved to support the explanations and the prediction that is mostly supported by the evidence is selected. This approach does not require any additional training or fine-tuning. Trivedi et al. (2022) propose an information retrieval chain-of-thought approach (IRCoT) which consists of interleaving retrieval with CoT for multi-step QA. The idea is to use retrieval to guide the CoT reasoning steps and conversely, using CoT reasoning to guide the retrieval step.

In all these works, a `retriever` is systematically called for every query in order to get the corresponding documents to augment the LM. These approaches also assume that the intent is contained in the query. The query could be augmented with the user's intent by providing a natural language description of the search task (instruction) in order to disambiguate the intent, as proposed by Asai et al. (2022). Also, the LM could query the retriever only occasionally—when a prompt suggests it to do so—which is discussed in the next subsection.

3.2.2 Querying search engines

In the previous paragraph, the information retrieval query corresponds to the LM's context. However, the LM can also have the ability to generate a query based on the prompt, thus enlarging its action space and becoming more active.

LaMDA is one example of an agent-like LM designed for dialogue applications. The authors pre-train the model on dialog data as well as other public web documents. In addition to this, to ensure that the model

is factually grounded as well as enhancing its conversational abilities, it is augmented with `retrieval`, a `calculator`, and a `translator` (Thoppilan et al., 2022). Furthermore, to improve the model's safety, *LaMDA* is fine-tuned with annotated data. Another example is *BlenderBot* (Shuster et al., 2022b), where the LM decides to generate a query based on a prompt. In this case, the prompt corresponds to the instruction of calling the search engine tool. *BlenderBot* is capable of open-domain conversation, it has been deployed on a public website to further improve the model via continual learning with humans in the loop. Similarly, *ReAct* uses few-shot prompting to teach a LM how to use different tools such as `search` and `lookup` in Wikipedia, and `finish` to return the answer (Yao et al., 2022b). Similarly, Komeili et al. (2021); Shuster et al. (2022a) propose a model that learns to generate an internet search query based on the context, and then conditions on the search results to generate a response. *ReAct* interleaves reasoning and acting, allowing for greater synergy between the two and improved performance on both language and decision making tasks. *ReAct* performs well on a diverse set of language and decision making tasks such as question answering, fact verification, or web and home navigation.

In general, reasoning can improve decision making by making better inferences and predictions, while the ability to use external tools can improve reasoning by gathering additional information from knowledge bases or environments.

3.2.3 Searching and navigating the web

It is also possible to train agents that can navigate the open-ended internet in pursuit of specified goals such as searching information or buying items. For example, *WebGPT* (Nakano et al., 2021) is a LM-based agent which can interact with a custom text-based web-browsing environment in order to answer long-form questions. In contrast with other models that only learn how to query retrievers or search engines like *LaMDA* (Thoppilan et al., 2022) or *BlenderBot* (Shuster et al., 2022b), *WebGPT* learns to interact with a web-browser, which allows it to further refine the initial query or perform additional actions based on its interactions with the tool. More specifically, *WebGPT* can `search` the internet, `navigate` webpages, `follow` links, and `cite` sources (see Table 3 for the full list of available actions). By accessing the internet, the agent is able to enhance its question-answering abilities, even surpassing those of humans as determined by human evaluators. The best model is obtained by fine-tuning *GPT3* on human demonstrations, and then performing rejection sampling against a reward model trained to predict human preferences. Similarly, WebShop (Yao et al., 2022a) is a simulated e-commerce website where an agent has to find, customize, and purchase a product according to a given instruction. To accomplish this, the agent must understand and reason about noisy text, follow complex instructions, reformulate queries, navigate different types of webpages, take actions to collect additional information when needed, and make strategic decisions to achieve its goals. Both the observations and the actions are expressed in natural language, making the environment well-suited for LM-based agents. The agent consists of a LM fine-tuned with behavior cloning of human demonstrations (*i.e.*, question-human demonstration pairs) and reinforcement learning using a hard-coded reward function that verifies whether the purchased item matches the given description. While there are other works on web navigation and computer-control, most of them assume the typical human interface, that takes as input images of a computer screen and output keyboard commands in order to solve digital tasks (Shi et al., 2017; Gur et al., 2019; 2021; Toyama et al., 2021; Humphreys et al., 2022; Gur et al., 2022). Since our survey focuses on LM-based agents, we will not discuss these works in detail.

3.3 Computing via Symbolic Modules and Code Interpreters

Although recent LMs are able to correctly decompose many problems, they are still prone to errors when dealing with large numbers or performing complex arithmetics (Mishra et al., 2022b). For example, vanilla *GPT3* cannot perform out-of-distribution addition, *i.e.* addition on larger numbers than those seen during the training even when provided with examples with annotated steps (Qian et al., 2022). In the context of reinforcement learning, the action space of a transformer agent is equipped with symbolic modules to perform *e.g.* arithmetic or navigation in Wang et al. (2022b). *Mind's Eye* (Liu et al., 2022b) invokes a `physics engine` to ground LMs physical reasoning. More precisely, a text-to-code LM is used to produce rendering code for the physics engine. The outcome of the simulation that is relevant to answer the question is then

appended in natural language form to the LM prompt. As a result, *Mind's Eye* is able to outperform the largest LMs on some specific physical reasoning tasks while having two order of magnitude less parameters.

For reasoning graph generation, *CoCoGen* (Madaan et al., 2022) propose to generate python code generating a graph instead of a serialized version of the graph. *PAL* (Gao et al., 2022) relies on CoT prompting of large LMs to decompose symbolic reasoning, mathematical reasoning, or algorithmic tasks into intermediate steps along with python code for each step (see Figure 6). The python steps are then offloaded to a `python interpreter` outputting the final result. They outperform CoT prompting on several benchmarks, especially on GSM-HARD, a version of GSM8K with larger numbers. See Table 1 for a comparison between *PAL* and other models on GSM8K. Similarly, Drori et al. (2022); Chen et al. (2022b) prompts *Codex* (Chen et al., 2021) to generate executable code-based solutions to university-level problems, math word problems, or financial QA. For code generation, Shi et al. (2022) execute several sampled generation on a small number of test inputs, and use the output to select a solution. Instead, Mishra et al. (2022a) extends existing datasets with solutions written as python programs. Then, they finetune a model to generate python solutions to mathematical reasoning problems. In the context of theorem proving, Wu et al. (2022c) uses large LMs to automatically formalize informal mathematical competition problem statements in Isabelle or HOL. Jiang et al. (2022) generate formal proof sketches, which are then fed to a prover.

Question: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
Answer: Roger started with 5 balls.

```
tennis_balls = 5
```

2 cans of 3 tennis balls each is

```
bought_balls = 2 * 3
```

tennis balls. The answer is

```
answer = tennis_balls * bought_balls
```

Question: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Answer:
<LM>

Figure 6: An example of few-shot PAL (Gao et al., 2022) prompt. **<LM>** denotes call to the LM with the above prompt. The prompts are based on the chain-of-thoughts prompting shown on Figure 1, and the parts taken from it are highlighted in green. In PAL, the prompts also contain `executable python code`, which performs operations and stores the results in the `answer` variable. When prompted with a new question, PAL generates a mix of executable code and explanation. The answer is obtained by executing the code and `print(answer)`.

3.4 Acting on the virtual and physical world

While the previous tools gather external information in order to improve the LM's predictions or performance on a given task, other tools allow the LM to act on the virtual or physical world. In order to do this, the LM needs to ground itself in the real-world by learning about affordances i.e. what actions are possible in a given state, and their effect on the world.

Controlling Virtual Agents. Recent works demonstrated the ability of LMs to control virtual agents in simulated 2D and 3D environments by outputting functions which can then be executed by computers in the corresponding environment, be it a simulation or the real-world. For example, Li et al. (2022b) fine-tune a pre-trained *GPT2* (Radford et al., 2019) on sequential decision-making problems by representing the goals and observations as a sequence of embeddings and predicting the next action. This framework enables strong

combinatorial generalization across different domains including a simulated household environment. This suggests that LMs can produce representations that are useful for modeling not only language but also sequential goals and plans, so that they can improve learning and generalization on tasks that go beyond language processing. Similarly, Huang et al. (2022a) investigate whether it is possible to use the world knowledge captured by LMs to take specific actions in response to high-level tasks written in natural language such as "make breakfast". This work was the first to demonstrate that if the LM is large enough and correctly prompted, it can break down high-level tasks into a series of simple commands without additional training. However, the agent has access to a predetermined set of actions, so not all natural language commands can be executed in the environment. To address this issue, the authors propose to map the commands suggested by the LM into feasible actions for the agent using the cosine similarity function. The approach is evaluated in a virtual household environment and displays an improvement in the ability to execute tasks compared to using the plans generated by the LM without the additional mapping. While these works have demonstrated the usefulness of LMs for controlling virtual robots, the following paragraph cover works on physical robots. Zeng et al. (2022) combine a LM with a visual-language model (VLM) and a pre-trained language-conditioned policy for controlling a simulated robotic arm. The LM is used as a multi-step planner to break down a high-level task into subgoals, while the VLM is used to describe the objects in the scene. Both are passed to the policy which then executes actions according to the specified goal and observed state of the world. Dasgupta et al. (2023) use 7B and 70B *Chinchilla* as planners for an agent that acts and observes the result in a PycoLab environment. Additionally, a reporter module converts actions and observations from pixel to text space. Finally, the agent in Carta et al. (2023) uses a LM to generate action policies for text-based tasks. Interactively learning via online RL allows to ground the LM internal representations to the environment, thus partly departing from the knowledge about statistical surface structure of text that was acquired during pre-training.

Command	Effect
`search <query>`	Send <query> to the Bing API and display a search results page
`clicked on link <link ID>`	Follow the link with the given ID to a new page
`find in page: <text>`	Find the next occurrence of <text> and scroll to it
`quote: <text>`	If <text> is found in the current page, add it as a reference
`scrolled down <1, 2, 3>`	Scroll down a number of times
`scrolled up <1, 2, 3>`	Scroll up a number of times
`Top`	Scroll to the top of the page
`back`	Go to the previous page
`end: answer`	End browsing and move to answering phase
`end: <nonsense, controversial>`	End browsing and skip answering phase

Table 3: The actions *WebGPT* can perform, taken from Nakano et al. (2021).

Controlling Physical Robots. Liang et al. (2022) use a LM to write robot policy code given natural language commands by prompting the model with a few demonstrations. By combining classic logic structures and referencing external libraries, e.g., for arithmetic operations, LMs can create policies that exhibit spatial-geometric reasoning, generalize to new instructions, and provide precise values for ambiguous descriptions. The effectiveness of the approach is demonstrated on multiple real robot platforms. LMs encode common sense knowledge about the world which can be useful in getting robots to follow complex high-level instructions expressed in natural language. However, they lack contextual grounding which makes it difficult to use them for decision making in the real-world since they do not know what actions are feasible in a particular situation. To mitigate this problem, Ahn et al. (2022) propose to teach the robot a number of low-level skills (such as "find a sponge", "pick up the apple", "go to the kitchen") and learn to predict how feasible they are at any given state. Then, the LM can be used to split complex high-level instructions into simpler subgoals from the robot's repertoire. The LM can then select the most valuable yet feasible skills for the robot to perform. This way, the robot can use its physical abilities to carry out the LM's instructions, while the LM provides semantic knowledge about the task. The authors test their approach, called *SayCan*, on various real-world tasks and find that it can successfully complete long, abstract instructions in a variety of

environments. To address the grounding problem, Chen et al. (2022a) propose *NLMap-SayCan*, a framework to gather and integrate contextual information into LM planners. *NLMap* uses a Visual Language Model (VLM) to create an open-vocabulary queryable scene representation before generating a context-conditioned plan. An alternative way of incorporating contextual information into the agent's decisions is to utilize linguistic feedback from the environment such as success detection, object recognition, scene description, or human interaction (Huang et al., 2022b). This results in improved performance on robotic control tasks such as table top rearrangement and mobile manipulation in a real kitchen. Finally, *RT-1* (Brohan et al., 2022) leverages large-scale, diverse, task-agnostic robotic datasets to learn a model that can follow over 700 natural language instructions, as well as generalize to new tasks, environments, and objects. *RT-1* makes use of *DIAL* (Xiao et al., 2022), an approach for automatically labeling robot demonstrations with linguistic labels via the vision-language alignment model *CLIP* (Radford et al., 2019).

4 Learning to reason, use tools, and act

The previous sections reviewed *what* LMs can be augmented with in order to endow them with reasoning and tools. We will now present approaches on *how* to teach them such abilities.

4.1 Supervision

A straightforward way of teaching LMs both to reason and to act consists in providing them with human-written demonstrations of the desired behaviours. Common ways of doing so are (i) via few-shot prompting as first suggested by Brown et al. (2020), where the LM is provided a few examples as additional context during inference, but no parameter updates are performed, or (ii) via regular gradient-based learning. Typically, supervised learning is done *after* an initial pre-training with a language modeling objective (Ouyang et al., 2022; Chung et al., 2022); an exception to this is recent work by Taylor et al. (2022), who propose to mix pre-training texts with human-annotated examples containing some form of explicit reasoning, marked with a special token. Some authors use supervised fine-tuning as an intermediate step, followed by reinforcement learning from human feedback (Nakano et al., 2021; Ouyang et al., 2022); see Section 4.2 for an in-depth discussion of such methods.

Few-shot prompting. Providing LMs with a few human-written *in-context* demonstrations of a desired behaviour is a common approach both for teaching them to reason (Wei et al., 2022c;b; Suzgun et al., 2022; Press et al., 2022) and for teaching them to use tools and act (Gao et al., 2022; Lazaridou et al., 2022; Yao et al., 2022b). This is mainly due to its ease of use: few-shot prompting only requires a handful of manually labeled examples and enables very fast experimentation as no model fine-tuning is required; moreover, it enables reusing the very same model for different reasoning tasks and tools, just by changing the provided prompt (Brown et al., 2020; Wei et al., 2022c). On the other hand, the ability to perform reasoning with chain-of-thoughts from a few in-context examples only emerges as models reach a certain size (Wei et al., 2022b; Chung et al., 2022), and performance depends heavily on the format in which examples are presented (Jiang et al., 2020; Min et al., 2022), the choice of few-shot examples, and the order in which they are presented (Kumar and Talukdar, 2021; Lu et al., 2022b; Zhou et al., 2022). Another issue is that the amount of supervision that can be provided is limited by the number of examples that fit into the LM's context window; this is especially relevant if (i) a new behaviour is so difficult to learn that it requires more than a handful of examples, or (ii) we have a large space of possible actions that we want a model to learn. Beyond that, as no weight updates are performed, the LM's reasoning and acting abilities are tied entirely to the provided prompt; removing it also removes these abilities.

Fine-tuning. As an alternative to few-shot prompting, the reasoning and acting abilities of a pre-trained LM can also be elicited by updating its parameters with standard supervised learning. This approach has been used both for teaching models to use tools, including search engines (Komeili et al., 2021; Shuster et al., 2022b), web browsers (Nakano et al., 2021), calculators and translation systems (Thoppilan et al., 2022), and for improving reasoning abilities (Chung et al., 2022). For the latter, examples of reasoning are typically used in the larger context of *instruction tuning* (Mishra et al., 2021; Sanh et al., 2022; Wang et al., 2022d; Ouyang et al., 2022), where, more generally, an LM's ability to follow instructions is improved based on human-labeled

examples. Examples are typically collected from crowd workers. In some cases, they can instead be obtained automatically: Nye et al. (2021) use execution traces as a form of supervision for reasoning, while Andor et al. (2019) use heuristics to collect supervised data for teaching a language model to use a calculator.

Prompt pre-training. A potential risk of finetuning *after* the pre-training phase is that the LM might deviate far from the original distribution and overfit the distribution of the examples provided during fine-tuning. To alleviate this issue, Taylor et al. (2022) propose to mix pre-training data with labeled demonstrations of reasoning, similar to how earlier work mixes pre-training data with examples from various downstream tasks (Raffel et al., 2020); however, the exact gains from this mixing, compared to having a separate fine-tuning stage, have not yet been empirically studied. With a similar goal in mind, Ouyang et al. (2022) and Iyer et al. (2022) include examples from pre-training during the fine-tuning stage.

Bootstrapping. As an alternative to standard fine-tuning, several authors propose to use *bootstrapping* techniques (e.g. Yarowsky, 1995; Brin, 1999) to leverage some form of indirect supervision. This typically works by prompting a LM to reason or act in a few-shot setup followed by a final prediction; examples for which the actions or reasoning steps performed did *not* lead to a correct final prediction are then discarded. For example, STaR (Zelikman et al., 2022) prompts a model to generate chain-of-thought reasoning sequences in a common sense question answering setup, but only keeps those chains that lead to the correct final answer for a given question. Finally, either the original LM or another (typically smaller) model is fine-tuned on all correct examples. As such, bootstrapping combines the data efficiency of few-shot prompting with some of the advantages of fine-tuning and can be successfully applied both to teach models to reason (Shridhar et al., 2022) and to use tools (Parisi et al., 2022).

4.2 Reinforcement learning

Supervised learning from human-created prompts is effective to teach models to reason and act. However, such data is difficult and costly to obtain. Human preference data — such as rankings or likes/dislikes — is much easier, faster, and cheaper to obtain than full demonstrations. For instance, it might be easier for a human to evaluate the quality of a summary than write one from scratch. Going further, it might be even easier to rank different summaries than scoring each separately. Such data cannot be used in a supervised setting, but can provide rewards in the context of Reinforcement Learning (RL) (Sutton and Barto, 2018).

RL has proven successful for learning complex behaviors through feedback-based interaction with an environment, and it has been used for applications such as playing games (Mnih et al., 2015; Silver et al., 2016; Vinyals et al., 2019; Team et al., 2021; Bakhtin et al., 2022) or controlling robots (Gu et al., 2017; Kalashnikov et al., 2018; Akkaya et al., 2019; Lee et al., 2020). When training a LM with RL, the LM can be considered an agent that learns a policy (i.e. a distribution over the model's vocabulary from which the next token is sampled) in order to optimize some reward function. Most of the existing work on RL and ALMs has focused on teaching LMs how to act rather than reason. The closest work on learning how to reason via RL is STaR (Zelikman et al., 2022), a bootstrapping-based approach that is discussed in Section 4.1

RL is a natural framework for training LMs to act and use tools since many of these tools are non-differentiable (e.g. search engines, calculators or programming language interpreters). Additionally, many tasks that benefit from interacting with tools resemble sequential decision making problems (e.g., navigating a web-browser to buy a specified product) and have a well-defined reward (e.g., 1 if the model buys the correct product and 0 otherwise). While there are early works focused on models that could interface with external tools, they employ ad-hoc tool-dependent architectures (Adolphs et al., 2022; Buck et al., 2018; Nogueira and Cho, 2017; Zhong et al., 2018). We do not cover them here since the main focus of our survey is instead on the acting and reasoning capabilities of standard general-purpose LM architectures trained with the language modeling objective.

Hard-coded reward functions. When teaching a LM how to use external tools, the standard practice is to update the weights of the model using a scalar reward generated by a hard-coded reward function. This task-dependent function is computed based on the tool output. The LM agent takes a textual input, which in RL terminology corresponds to the current state of the environment, and generates a sequence of tokens, or

actions in RL terms. Optimization is done through policy gradient algorithms like REINFORCE (Williams, 1992), PPO and similar variants (Schulman et al., 2017; Ramamurthy et al., 2022).

Initial works on training LMs to use tools via RL mostly focused on searching and fetching additional factual information. Common tools for such information-seeking tasks are `document retrievers`, `question answering systems`, and `search engines`. The first two consist in retrieving document from a pre-defined set of text documents, or in retrieving an answer based on some input query. However, a search engine allows for more structured interactive search where, for instance, the model further refines the initial query or performs additional actions based on the initial output of the tool. For example, Wu et al. (2022d) perform conversational question-answering by teaching a LM via RL to rewrite queries in order to feed them to an off-the-shelf retriever. The reward function is a contrastive retrieval-accuracy metric based on the token overlap between following conversation rounds and retrieved passages. Another example is the work from Liu et al. (2022a): *RAINIER* is a LM able to generate contextually relevant questions that are optimized to query a frozen `QA system`. After distilling knowledge from a larger *GPT3* (Brown et al., 2020) model into a smaller *T5* model (Raffel et al., 2020), *RAINIER* is finetuned using PPO (Schulman et al., 2017) with feedback provided by the pre-trained question answering model from Khashabi et al. (2020). Interestingly, this work is an example of a LM learning to use another frozen neural model as an external tool.

Yao et al. (2022a) use RL to teach a language model to navigate a `virtual shop` and buy items constrained on attributes like color and price. Similar to *WebGPT* (Nakano et al., 2021), the model is given a goal in textual format and allowed to perform a limited set of actions. Prompted with a user-generated instruction, in a multi-task learning setup, the model needs to simultaneously understand the query and browse the web to search for the right product. The reward is a hard-coded text-matching function based on the similarity between the model-purchased written description of the item and the given shopping instruction. Optimization is performed with the A3C algorithm (Mnih et al., 2016), a variant of the standard actor-critic method. While the model still lags behind human experts, they found that fine-tuning with RL after training on human demonstrations improves performance. This provides additional evidence of the benefits of reward-based learning for endowing LMs with the ability to interact with external tools.

While interacting with a `search engine` or a `document retriever` allows a model to augment its current context with additional input, it is often necessary to process structured information when interacting with tools like a `knowledge base`. Dognin et al. (2021) train a LM to learn how to interface with a graph-based knowledge base by performing the text2graph and graph2text tasks. The model, based on a *T5* architecture (Raffel et al., 2020) and trained with the vanilla policy gradient algorithm REINFORCE (Williams, 1992), can perform bidirectional generation of text and graphs and shows state-of-the-art performance on tasks related to knowledge base automated construction from text and vice versa. The *T5*-based agent is trained to directly maximize graph2text metrics such as BLEU (Papineni et al., 2002a), METEOR (Banerjee and Lavie, 2005), and chrF++ (Popović, 2017), or text2graph ones such as F1, Precision, and Recall.

Finally, it is also possible to leverage RL at inference time only. For example, Cao et al. (2023) propose a method to avoid generating tokens that will likely lead to toxic content by framing it as a RL problem. More precisely, a reward model and a value function are trained to evaluate respectively the toxicity of a sentence and the probability of the so-far generated tokens to lead to a toxic generation. At inference time, the latter is used to truncate the next token probability distribution towards the tokens that are less likely to lead to a toxic generation.

Human feedback. Evaluating the quality of machine-generated text is non-trivial because it can vary depending on the context, individual preferences, and user's intentions. For example, in some contexts, a user might require creative writing, while in others it may just require factual information. Model outputs should be judged accordingly and should be able to capture such intent differences. Several metrics based on heuristics like BLEU (Papineni et al., 2002b) and ROUGE (Lin, 2004) have been developed for comparing model outputs to reference texts. However, they fail to fully capture the quality of generations with respect to human intentions. Human feedback can be exploited to improve the quality of machine-generated text, for example for dialog agents (Xu et al., 2022). In particular, Reinforcement Learning from Human Feedback (RLHF) (Knox and Stone, 2008; MacGlashan et al., 2017; Christiano et al., 2017; Warnell et al., 2018) aims to overcome these limitations by using human preferences as an evaluation metric and as an objective function

to optimize the language model. Using RLHF allows LMs to be more closely aligned with complex human preferences and values which are difficult to capture by hard-coded reward functions.

RLHF works by using a pre-trained LM to generate text, which is then evaluated by humans by, for example, ranking two model generations for the same prompt. This data is then collected to learn a reward model that predicts a scalar reward given any generated text. The reward captures human preferences when judging model output. Finally, the LM is optimized against such reward model using RL policy gradient algorithms like PPO (Schulman et al., 2017). RLHF can be applied directly on top of a general-purpose LM pre-trained via self-supervised learning. However, for more complex tasks, the model's generations may not be good enough. In such cases, RLHF is typically applied after an initial supervised fine-tuning phase using a small number of expert demonstrations for the corresponding downstream task (Ramamurthy et al., 2022; Ouyang et al., 2022; Stiennon et al., 2020).

A successful example of RLHF used to teach a LM to use an external tool stems from *WebGPT* (Nakano et al., 2021), discussed in 3.2.3, a model capable of answering questions using a `search engine` and providing references to support such answers. The tool interface is a simplified text-based web-browser. The model architecture is based on *GPT3* (Brown et al., 2020) and is trained to perform browsing actions expressed in natural language. The model is fine-tuned on question-human demonstration pairs, before further optimization via RLHF. On two QA datasets, *WebGPT*'s answers are preferred relative to human-generated ones and tend to be more factual than the original vanilla *GPT3* model. Similarly, Menick et al. (2022) propose *GopherCite*, a *Gopher*-based LM model (Rae et al., 2021) fine-tuned with RLHF that can cite supporting evidence when answering questions and abstain from answering when unsure. In contrast with *WebGPT*, *GopherCite* uses an information retrieval external module rather than a web-browser to find relevant information that improves its question answering capabilities. Besides learning to use external tools, RLHF has also proven useful for a wide range of language generation tasks, from summarization (Ziegler et al., 2019; Wu et al., 2021; Stiennon et al., 2020) to training more helpful, harmless, and accurate assistants (Glaese et al., 2022; Cohen et al., 2022; Ouyang et al., 2022; Bai et al., 2022). Since these works do not focus on training models to reason and act, they are out of the scope of this survey.

4.3 Limitations and future directions

Despite recent algorithmic progress and performance improvements, current RL methods still suffer from instability issues which can make training difficult and slow (Ramamurthy et al., 2022; Snell et al., 2022). While supervised learning has been an efficient and robust way to fine-tune language models on specific tasks (Mishra et al., 2021; Sanh et al., 2022; Wang et al., 2022b), this assumes the existence of a large number of expert demonstrations, which can be difficult and costly to obtain. This is particularly true for tasks that require reasoning and acting where we do not have readily available data. A possible solution to the lack of quality data problem could come from bootstrapping methods and offline RL. The promise of these methods is that a dataset can be generated via feedback and interactions of any behavior policy and can be used to train an improved policy. Therefore, combining a data-driven approach with offline RL could give the "best of both worlds": learning from counterfactual events in a scalable and stable way (Levine et al., 2020). In the offline regime, though, there are some open challenges. The maximum improvement can be limited by a number of factors such as: the suboptimality of the initial behavior policy, the dimensionality of the state and the action space, the length of the effective horizon, the accumulation of errors and distributional shifts due to the discrepancy between the behavior policy and the learned one (Levine et al., 2020). Recent works (Zelikman et al., 2022; Snell et al., 2022) have shown that such approaches could reach performance that goes beyond that of the expert demonstrations or improve over initial model generations. For example, Snell et al. (2022) introduce a new offline RL algorithm called ILQL which learns from a static dataset of demonstrations and their associated rewards by estimating a value function and using it to optimize LM generations. ILQL combines online RL flexible optimization framework with the simplicity and ability to learn from existing datasets of supervised learning, resulting in good performance on dialogue tasks. As explained in Section 4, Zelikman et al. (2022) employ a bootstrapping approach for teaching LMs to reason, which can be seen as an approximation to policy gradient algorithms.

Recently, Schick et al. (2023) proposed *Toolformer*, a model that teaches itself to use tools in a self-supervised way. This is achieved by first using the few-shot abilities of an existing LM to sample a large amount of

potential tool uses. For instance, the model can call a calculator API to augment its context, e.g., "*Out of 1400 participants, 400 (or [Calculator(400 / 1400)→ 0.29] 29% passed the test.*" Then, the model is fine-tuned on its own generations, filtering them based on whether they reduce perplexity for future tokens generations. This method enables using several tools (e.g., a `calendar`, a `calculator`, or an `information retrieval system`). However, it was tested in a limited setup of using a single tool at once, since examples of tool use were independently sampled. We believe that studying how this approach could be extended to more complex multi-step tool uses is a promising research direction for a generalist LM-based agent.

5 Discussion

Moving away from language modeling. Is a model trained to do intermediate reasoning steps or having access to the internet still purely performing language modeling? Indeed, in NLP, language modeling (Bahl et al., 1983) is generally defined as the task of predicting missing tokens given a context and is relied heavily on for pre-training models. However, several techniques have been developed to later fine-tune models (Ziegler et al., 2019; Wei et al., 2022a; Sanh et al., 2022) to perform various natural language tasks, which could be seen as moving away from traditional language modeling. In particular, the texts used to fine-tune LMs are not just found on the internet, but rather designed to explicitly inject some level of grounding. One of the argument advocated recently in Goldberg (2023) is that "*it might be much easier to learn from direct instructions like these than it is to learn from non-instruction data*". This argument can be supported by the recent work of Giannou et al. (2023), showing both theoretically and in practice that even shallow looped transformers can follow instructions and be programmed as general purpose computers. Intuitively, a text is the result of complex intermediate thoughts that are hidden. Therefore, the superficial text used for supervision can be seen as representing only the logs of these thoughts, thus lacking of context. Conversely, with task-oriented supervised data, we can explicitly ground the answer with the intermediate steps. In this regard, the resulting model may not be considered as a language model. And yet, the task is still about predicting the next token given text only. The argument is all the more true for ALMs since they can augment their context.

In particular, tool-augmented LMs might actually lose the ability to assign a probability to the next token - which is at the core of language modeling: whereas a regular LM can easily compute $p(x_t \mid x_1, \ldots, x_{t-1})$, a tool-augmented LM has to consider all possible tool uses, e.g. $p(x_t \mid x_1, \ldots, x_{t-1}) = \sum_c p(c \mid x_1, \ldots, x_{t-1}) \cdot p(x_t \mid x_1, \ldots, x_{t-1}, c)$ where c is a tool, which might not be tractable. First, in the general case, marginalizing over all the tools is not enough, one also has to marginalize over all possible tool use. For many tools however, the probability $p(c \mid x_1, \ldots, x_{t-1})$ associated with each possible tool output cannot be modeled. Take for example web browsing: for a fixed query, the result may vary day to day unpredictably as both the internet and the search algorithms are constantly evolving, making the evaluation of $p(c \mid x_1, \ldots, x_{t-1})$ impossible unless the tool uses a past snapshot of the internet, which is generally not wanted. Second, in the case where a tool can be called within another tool call, the number of tool uses can become exponential in the number of tools. For example, within a tool call, an ALM could browse the web by reading the content of a web page, which is text, then apply another of the tools to the text it found to refine its query, etc. If for some reason we require a depth that is equal to the number of tools, we have an exponential complexity.

For these reasons, we refer to Augmented Language Models (ALMs) in this survey, to distinguish from Language Modeling in the traditional sense.

A tradeoff between memorizing and querying tools. Is it preferable to memorize information in the model weights, or to leverage external tools? Some situations arguably require external tools, for example computing 213443^{344}. However, many information are well known facts such as "The Eiffel tower is located in Paris" or $1 + 2 = 3$, and should not be offloaded. And, when learning world representations, memorization is not only desirable, but also deeply connected to reasoning (Hayes et al., 2014). Can ALMs be calibrated enough to decide when and when not to use a tool? Could a computation budget for each tool be integrated into the loss to let the model learn to do so?

Generalizing the non-parametric framework. A motivation behind information retrieval augmented LMs such as *RETRO* (Borgeaud et al., 2022) and *Atlas* (Izacard et al., 2022) is to develop a class of LM

requiring less parameters through relying on an external non-parametric memory. The motivation for using other kind of tools such as `code interpreter` or `calculator` has been slightly different so far: for instance, Cobbe et al. (2021) use a calculator to improve accuracy on tasks requiring arithmetic. Yet, the paradigm of tool-augmented LMs can be seen as a generalization of the non-parametric framework. Indeed, beyond information retrieval, LMs can delegate any kind of abilities such as calculus to the corresponding external tools. By avoiding to store rarely accessed knowledge in their weights, tool-augmented LMs may have better scaling laws and thus yield smaller models retaining the capabilities of their largest counterpart. Combined with the possibility to access recent information from the external world thus avoiding frequent updates, non-parametric generalization holds great benefits for ALMs.

A path towards autonomous machine intelligence? A concept for an autonomous intelligent agent was proposed by LeCun (2022). We now discuss to what extent ALMs instantiate this idea. In LeCun (2022), the agent is composed of different modules starting from a world model and a short-term memory. Essentially, the agent takes actions via an actor module based on its world model, perception module, and short-term memory so as to minimize some cost. The agent is also equipped with a configurator module for modulating the world model, the perception, the actor and the cost given the task at hand.

Translating into this framework, the ALM's weights essentially contain the world model, perception and actor modules. The short-term memory can be identified with the ALM's context or prompt. Based on its perception of the context and its world model, the ALM would take actions by outputting special tokens, and perceive the result. The configurator module remains elusive but may be implicit: it can be seen as the conditioning induced by the ALM's context, for example an initial prompt such as "You are a kind and helpful assistant". Finally, the cost remains fixed in this framework, and could be the ALM's perplexity mixed with a computational cost associated to reasoning and using external tools.

However, an important feature of the agent in LeCun (2022) is its ability to plan, defined by the decomposition of a complex task into subtasks: in the ALM's context, planning is akin to reasoning, a slight abuse of terminology as it is not clear whether LMs reason as humans do as noted in Section 2. LeCun (2022) propose to implement reasoning (under the term planning) as the minimization of an energy with respect to a hierarchical combination of actions. Since ALMs only perform predictions at the token level, they cannot reason according to LeCun (2022)'s view and may be still limited to System 1 tasks, *i.e.* that rely on reflex rather than logic and thinking. Whether System 2, *i.e.* the opposite abilities can be obtained by pushing current methods remains uncertain. For example, LMs are deprived from global consistency beyond their maximum sequence length: as an illustration, two different discussions with the same LM will result in inconsistencies. This is a strong limitation when it comes to solving complex problems that require to perform a large number of sub-goals such as writing a research paper, where one has an initial mental state that includes the current results and the angle of the paper. This process is not linear and results from different interactions, e.g., new ideas while reading some related works. The mental state is maintained although updated trough all the process, such that we keep in mind the big picture. Although more compute and larger input size could mitigate the issue, another solution may be to endow LMs with adequate components. In this regard, a model architecture that intrinsically makes the LM consistent with an energy function as suggested in LeCun (2022) could constitute a promising venue.

Finally, our survey sees LMs as the central piece of a generalist agent that could reason in natural language and interact with external tools. Along these lines, Wang et al. (2023) uses a LM as a centralized planner to generate goal sequences for solving tasks in the game of Minecraft. Through a feedback loop and intermediate checks on subgoals execution, the LM can explain mistakes of the goal executor and refine its original plan. However, we note that a LM-based controller might not be the only viable approach for a generalist agent. Recent work on the game of Diplomacy (Bakhtin et al., 2022), a long-standing challenge for AI agents due to its complex planning and reasoning dynamics, employs an ad-hoc planning model trained via self-play and reinforcement learning. Here the LM is used to interact with other players, thus as an external communication module grounded in the current state of the game. This offers an alternative view of LMs as agents specialized to communicate with humans, albeit in the restricted setting of a Diplomacy game. We believe that (A)LMs will play a central role in the next generation of powerful interactive systems, whether as centralized controller

of a modular system or as a language-only module that needs to interact with an orchestrator remains an open research question.

Augmented Language Models benefits. Overall, ALMs offer many potential advantages over traditional LMs.

- *Truthfulness*: As the current LM's training objective is arguably responsible for inciting the generation of seemingly plausible but not factual information, grounding the predictions through some tools should lead to more trustworthy models. However, although this conclusion is straightforward when equipping a LM with a calculator, there is surprisingly little evidence of it for information retrieval augmented LMs (Krishna et al., 2021). One of the reasons is the presence of a lot of non-truthful information in the web. Investigating this direction will be critical for making LM reliable.

- *Estimating and reducing uncertainty*: Extending the maximum-likelihood paradigm by letting the model reason and access additional information could help models to learn what they know and what they don't. Some papers suggest that LMs are already well calibrated (Kadavath et al., 2022), i.e. there is a high correlation between the accuracy of their predictions and the corresponding likelihood. This uncertainty could be directly exploited by ALMs to know when to rely on their own weights, or when to query an external tool.

- *Interpretability*: Deep learning models are often considered to be black boxes, and their predictions are difficult to interpret. Providing intermediate reasoning steps and relying on tools should help to make ALMs more interpretable. In particular, we can expect that being able to cite the sources used to compose the answer to be critical. However, some works Lewkowycz et al. (2022) pointed out that chain-of-thoughts can lead to the correct predictions even though the intermediate reasoning doesn't make any sense, indicating clear challenges for researchers exploring this direction.

- *Enhanced capabilities*: ALMs with improved reasoning abilities and tools can be more helpful assistants and solve a wider range of tasks than standard LMs. For example, an ALM connected to a python interpreter can run code and experiments on a user's behalf, which a vanilla LM cannot do. In addition, a feedback loop can emerge between reasoning and acting, where each ability further improves the other (Yao et al., 2022b). Interacting with external tools, entities, and environments can improve reasoning since it allows the ALM to collect additional information and ground itself in the real-world. Similarly, reasoning can improve the ALM's decision making abilities such as when and how to use a certain tool.

Cost of using tools. To the best of our knowledge, the cost of using tools has not yet been taken into account comprehensively.

Overall, using tools seem to be beneficial in terms of energy use. Retrieval-augmented language models are a good example of a more efficient way to handle new information since updating an external data store can be orders of magnitude cheaper than re-training a LLM for scratch every time we get new data (as argued for example in Borgeaud et al. (2022)). This stays true for using a calculator or browsing the web for example. One can therefore assume that relying on tools rather than storing knowledge in parameters is generally more energy-efficient than training, frequently updating a LLM, and scaling it if some ability remains out of reach, if one assumes an optimal tool usage (i.e., not using each tool at each token). Then, how to assess the cost of using a tool versus another?

We believe that the most natural way of estimating such cost is to consider two factors: (i) the price per tool request, and (ii) the time required to fulfill the request. These factors make sense from the perspective of the ALM, and are a reasonable way to take into account the cost of creating and maintaining the tool since this should typically be reflected in the pricing. Interestingly, including such cost at training and inference time under the form of a loss term for example could help ALMs to develop optimal tool use, and lead to the emergence of models that know how to balance between tool use or internal chain of thoughts.

Ethical concerns. ALMs raise new ethical concerns. LM predictions based on tools may look more trustworthy and authoritative at a first glance, when in fact many of them will still be incorrect. Moreover, we can expect this phenomenon to be amplified as LMs reason in quite a similar manner to humans (Dasgupta et al., 2022), making it even harder to detect mistakes. Hence, ALMs may be leveraged to generate more convincing fake news and conspiracy theories. Conversely, conditioning ALMs on "safe" and trusted sources should lead to more factuality and less toxicity in their answers.

While such ethical concerns apply to most tools, it is important to distinguish between passive and active tools. The former only collects external information and passes it to the LM's context, while the latter allows it to act on the virtual or physical world without human validation in the loop. An example of active tool is letting the LM control a search engine. Hence, there exists a broad spectrum of possible harmful consequences of LM usage. We are moving from passive LMs that generate text in isolation of the external environment, towards ALMs that act in the real world. In this context, the aforementioned ethical concerns may resonate even further, as ALM will be connected to more and more tools and environments.

Currently, training LLM results in a large number of gas emissions, leveraging tools may result in smaller hence cheaper models that are more environmentally friendly both at training, inference, and update. ALMs are therefore a promising avenue to decrease the environmental impact of LLMs.

ALMs also have the potential to transform the LM landscape. Nowadays, state of the art LLMs have been the product of large research organizations, fine-tuning these models towards more versatility (via reasoning and learning to use different tools) and agency (via actions) may be within range of smaller entities and perhaps individuals. We may therefore see the burgeoning of a market of open, possibly specialized and even personalized ALM agents, learning to use new tools, handling various data modalities and interacting with each other. We may also see the emergence of a new breed of app store, dedicated (via appropriate documentation or API constraints for example) to be used by ALMs. Overall, this new ecosystem may further accelerate the current pace of LLM research and application to the real world, while also making its control even more uncertain.

6 Conclusion

This survey presented works in which LMs are augmented with better reasoning and tools. In most of the works, LMs augment their context with additional relevant information useful to perform missing token prediction. As many of these augmentations are non-parametric, *i.e.* involve calling an external, possibly non-parametric module, such LMs arguably depart from the classical language modeling paradigm, hence our decision to dub them Augmented Language Models. Although several works focus either on reasoning or acting skills of LMs, most of them rely on human annotation which may not be scalable, e.g., hand-crafted few-shot prompting, or reinforcement learning from human feedback. How to equip language models with meaningful augmentations in a fully self-supervised fashion remains an open research question. Additionally, as very few works combine reasoning and tools, future efforts should study the integration and fruitful interaction between these two skills. Overall, we believe studying Augmented Language Models is a promising and exciting research avenue towards the next generation of deep learning systems capable of complex and useful human-machine interaction.

References

Leonard Adolphs, Benjamin Boerschinger, Christian Buck, Michelle Chen Huebscher, Massimiliano Ciaramita, Lasse Espeholt, Thomas Hofmann, and Yannic Kilcher. Boosting search engines with interactive agents. *Transactions on Machine Learning Research (TMLR)*, 2022.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. Giving BERT a calculator: Finding operations and arguments with reading comprehension. 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.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hannaneh Hajishirzi. One question answering model for many languages with cross-lingual dense passage retrieval. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260*, 2022.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, 1983.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sandra Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David J. Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378:1067 – 1074, 2022.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72. Association for Computational Linguistics, 2005.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning (ICML)*, 2022.

Sergey Brin. Extracting patterns and relations from the world wide web. In *The World Wide Web and Databases*, pages 172–183. Springer Berlin Heidelberg, 1999.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

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. Language models are few-shot learners, 2020.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. Ask the right questions: Active question reformulation with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2018.

Meng Cao, Mehdi Fatemi, Jackie Chi Kit Cheung, and Samira Shabanian. Systematic rectification of language models via dead-end analysis. In *International Conference on Learning Representations (ICLR)*, 2023.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning, 2023.

Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. *arXiv preprint arXiv:2209.09874*, 2022a.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2022b.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *arXiv*, 2022.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*, 2017.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Deborah Cohen, Moonkyung Ryu, Yinlam Chow, Orgad Keller, Ido Greenberg, Avinatan Hassidim, Michael Fink, Yossi Matias, Idan Szpektor, Craig Boutilier, et al. Dynamic planning in open-ended dialogue using reinforcement learning. *arXiv preprint arXiv:2208.02294*, 2022.

Antonia Creswell and Murray Shanahan. Faithful reasoning using large language models. *arXiv preprint arXiv:2208.14271*, 2022.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

Ishita Dasgupta, Christine Kaeser-Chen, Kenneth Marino, Arun Ahuja, Sheila Babayan, Felix Hill, and Rob Fergus. Collaborating with language models for embodied reasoning, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Pierre L Dognin, Inkit Padhi, Igor Melnyk, and Payel Das. Regen: Reinforcement learning for text and knowledge base generation using pretrained language models. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Chris Donahue, Mina Lee, and Percy Liang. Enabling language models to fill in the blanks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, et al. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32), 2022.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. Compositional semantic parsing with large language models. *arXiv preprint arXiv:2209.15003*, 2022.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. Successive prompting for decomposing complex questions. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2022.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. *arXiv preprint arXiv:2301.13196*, 2023.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Yoav Goldberg. Some remarks on large language models, 2023. URL https://gist.github.com/yoavg/59d174608e92e845c8994ac2e234c8a9.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations (ICLR)*, 2017.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396, 2017.

Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tur. Learning to navigate the web. *International Conference on Learning Representations (ICLR)*, 2019.

Izzeddin Gur, Natasha Jaques, Kevin Malta, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. Adversarial environment generation for learning to navigate the web. *arXiv preprint arXiv:2103.01991*, 2021.

Izzeddin Gur, Ofir Nachum, Yingjie Miao, Mustafa Safdari, Austin Huang, Aakanksha Chowdhery, Sharan Narang, Noah Fiedel, and Aleksandra Faust. Understanding html with large language models. *arXiv preprint arXiv:2210.03945*, 2022.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International Conference on Machine Learning (ICML)*, 2020.

Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language models are general-purpose interfaces. *arXiv preprint arXiv:2206.06336*, 2022.

Brett K Hayes, Evan Heit, and Caren M Rotello. Memory, reasoning, and categorization: parallels and common mechanisms. *Frontiers in Psychology*, 5:529, 2014.

Hangfeng He, Hongming Zhang, and Dan Roth. Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*, 2022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022b.

Peter C Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy Lillicrap. A data-driven approach for learning to control computers. In *International Conference on Machine Learning (ICML)*, 2022.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Ves Stoyanov. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*, 2022.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022.

Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*, 2022.

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

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. arxiv e-prints, page. *arXiv preprint arXiv:1806.10293*, 2018.

Seyed Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. Lambadal backward chaining for automated reasoning in natural language, 2022.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2019.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*, 2020.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*, 2020.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

W Bradley Knox and Peter Stone. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE international conference on development and learning*, pages 292–297. IEEE, 2008.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. Internet-augmented dialogue generation. *ArXiv*, abs/2107.07566, 2021.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. Hurdles to progress in long-form question answering. *arXiv preprint arXiv:2103.06332*, 2021.

Sawan Kumar and Partha Talukdar. Reordering examples helps during priming-based few-shot learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4507–4518, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.395. URL https://aclanthology.org/2021.findings-acl.395.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR, 2018.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. Hypertree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering, 2022. URL https://arxiv.org/abs/2203.05115.

Yann LeCun. A path towards autonomous machine intelligence, 2022.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*, 2019.

Chris Lengerich, Gabriel Synnaeve, Amy Zhang, Hugh Leather, Kurt Shuster, François Charton, and Charysse Redwood. Contrastive distillation is a sample-efficient self-supervised loss policy for transfer learning. *arXiv preprint arXiv:2212.11353*, 2022.

Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022.

Belinda Li, Jane Yu, Madian Khabsa, Luke Zettlemoyer, Alon Halevy, and Jacob Andreas. Quantifying adaptability in pre-trained language models with 500 tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4696–4715, Seattle, United States, July 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.346. URL https://aclanthology.org/2022.naacl-main.346.

Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022b.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *arXiv preprint arXiv:2205.14217*, 2022c.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Jiacheng Liu, Skyler Hallinan, Ximing Lu, Pengfei He, Sean Welleck, Hannaneh Hajishirzi, and Yejin Choi. Rainier: Reinforced knowledge introspector for commonsense question answering. *arXiv preprint arXiv:2210.03078*, 2022a.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*, 2022b.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *arXiv preprint arXiv:2209.09513*, 2022a.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 04 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00369. URL https://doi.org/10.1162/tacl_a_00369.

James MacGlashan, Mark K Ho, Robert Loftin, Bei Peng, Guan Wang, David L Roberts, Matthew E Taylor, and Michael L Littman. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*, pages 2285–2294. PMLR, 2017.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. *arXiv preprint arXiv:2210.07128*, 2022.

John McCarthy et al. *Programs with common sense*. RLE and MIT computation center Cambridge, MA, USA, 1960.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*, 2022.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, 2019.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work?, 2022. URL https://arxiv.org/abs/2202.12837.

Swaroop Mishra and Elnaz Nouri. Help me think: A simple prompting strategy for non-experts to create customized content with models. *arXiv preprint arXiv:2208.08232*, 2022.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *arXiv preprint arXiv:2104.08773*, 2021.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, et al. Lila: A unified benchmark for mathematical reasoning. *arXiv preprint arXiv:2210.17517*, 2022a.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.246. URL https://aclanthology.org/2022.acl-long.246.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1928–1937. JMLR.org, 2016.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Rodrigo Nogueira and Kyunghyun Cho. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 574–583, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1061. URL https://aclanthology.org/D17-1061.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002a.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002b.

Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models. *arXiv preprint arXiv:2205.12255*, 2022.

Pruthvi Patel, Swaroop Mishra, Mihir Parmar, and Chitta Baral. Is a question decomposition unit all we need? *arXiv preprint arXiv:2205.12538*, 2022.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Maja Popović. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618. Association for Computational Linguistics, 2017.

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

Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. Limitations of language models in arithmetic and symbolic induction, 2022.

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

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

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2021.

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

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma 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, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations (ICLR)*, 2022.

Timo Schick, Jane Dwivedi-Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. Peer: A collaborative language model. *arXiv preprint arXiv:2208.11663*, 2022.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì†, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. Continual-t0: Progressively instructing 50+ tasks to language models without forgetting. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. Blank language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I Wang. Natural language to code translation with execution. *arXiv preprint arXiv:2204.11454*, 2022.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning (ICML)*, 2017.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. Replug: Retrieval-augmented black-box language models, 2023.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*, 2022.

Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion. *arXiv preprint arXiv:2203.13224*, 2022a.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Y-Lan Boureau, Melanie Kambadur, and Jason Weston. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*, 2022b.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. Androidenv: a reinforcement learning platform for android. *arXiv preprint arXiv:2105.13231*, 2021.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Boshi Wang, Xiang Deng, and Huan Sun. Iteratively prompt pre-trained language models for chain of thought. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022a.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Behavior cloned transformers are neurosymbolic reasoners. *arXiv preprint arXiv:2210.07382*, 2022b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022c.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. Super-natural instructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022d.

Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents, 2023. URL https://arxiv.org/abs/2302.01560.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 1, 2018.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research (TMLR)*, 2022b.

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

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations (ICLR)*, 2020.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–10, 2022a.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2022b.

Yuhuai Wu, Albert Q Jiang, Wenda Li, Markus N Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022c.

Zeqiu Wu, Yi Luan, Hannah Rashkin, David Reitter, and Gaurav Singh Tomar. Conqrr: Conversational query rewriting for retrieval with reinforcement learning. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022d.

Ted Xiao, Harris Chan, Pierre Sermanet, Ayzaan Wahid, Anthony Brohan, Karol Hausman, Sergey Levine, and Jonathan Tompson. Robotic skill acquisition via instruction augmentation with vision-language models. *arXiv preprint arXiv:2211.11736*, 2022.

Jing Xu, Megan Ung, Mojtaba Komeili, Kushal Arora, Y-Lan Boureau, and Jason Weston. Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback, 2022.

Jingfeng Yang, Haoming Jiang, Qingyu Yin, Danqing Zhang, Bing Yin, and Diyi Yang. Seqzero: Few-shot compositional semantic parsing with sequential prompts and zero-shot models. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022a.

Kevin Yang, Dan Klein, Nanyun Peng, and Yuandong Tian. Doc: Improving long story coherence with detailed outline control. *arXiv preprint arXiv:2212.10077*, 2022b.

Kevin Yang, Nanyun Peng, Yuandong Tian, and Dan Klein. Re3: Generating longer stories with recursive reprompting and revision. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022c.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022b.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 1995.

Ping Yu, Tianlu Wang, Olga Golovneva, Badr Alkhamissy, Gargi Ghosh, Mona Diab, and Asli Celikyilmaz. Alert: Adapting language models to reasoning tasks. *arXiv preprint arXiv:2212.08286*, 2022.

Eric Zelikman, Jesse Mu, Noah D Goodman, and Yuhuai Tony Wu. Star: Self-taught reasoner bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language, 2022. URL https://arxiv.org/abs/2204.00598.

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

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning, 2018. URL https://openreview.net/forum?id=Syx6bz-Ab.

Zexuan Zhong, Tao Lei, and Danqi Chen. Training language models with memory augmentation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

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

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.