ON THE CAPABILITIES AND RISKS OF LARGE LANGUAGE MODELS

BY

JIE HUANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Doctoral Committee:

Professor Kevin Chen-Chuan Chang, Chair
Assistant Professor Hao Peng
Associate Professor Hanghang Tong
Assistant Professor Tianyin Xu
Assistant Professor Diyi Yang, Stanford University

ABSTRACT

The advent of Large Language Models (LLMs) has significantly influenced the field of artificial intelligence, offering remarkable text generation capabilities through their vast number of parameters. These advancements have established new benchmarks across various domains. However, despite the impressive capabilities of LLMs, there exist critical limitations and ethical challenges. This dissertation critically examines the capabilities of LLMs, including their reasoning abilities, and explores potential risks, such as privacy leakage. Through this analysis, we underscore the crucial need to improve the capabilities of LLMs while mitigating the associated risks.

Based on this understanding, we propose methodologies to augment and safeguard LLMs. To enhance their functionality, we develop techniques to integrate LLMs with external knowledge and design an innovative data structure for knowledge representation. Additionally, we advocate for incorporating citation mechanisms within LLMs to promote transparency, accountability, and respect for intellectual property. Through rigorous research and the introduction of cutting-edge techniques, this dissertation aims to advance the capabilities of LLMs while ensuring their responsible and ethical use, ultimately contributing to the development of powerful and trustworthy artificial intelligence systems.

To the lovely world.

ACKNOWLEDGMENTS

I am deeply grateful for the support and guidance I have received throughout my Ph.D. journey. First, I would like to express my sincere gratitude to my advisor, Kevin Chen-Chuan Chang. Together, we initiated and led the NLP research in our group. Kevin's support and guidance have been crucial in shaping my research agenda. His rigorous research attitude and constant encouragement to pursue my interests have been pivotal in helping me become a well-established researcher.

I also extend my thanks to Wen-mei Hwu and Jinjun Xiong for their support and advice during the first two years of my Ph.D. at the IBM-Illinois Center for Cognitive Computing Systems Research. Additionally, I am grateful to Hao Peng, Hanghang Tong, Tianyin Xu, and Diyi Yang for serving on my dissertation committee and providing insightful comments and suggestions on my thesis research.

During my Ph.D., I have been fortunate to intern at several outstanding industrial research labs. I would like to extend my heartfelt thanks to my mentors: Xinying Song and Denny Zhou at Google DeepMind, Wei Ping at NVIDIA, and Yifan Gao at Amazon, for hosting my internships and providing me with tremendous support and guidance. I am also especially thankful to Xinyun Chen and Chen Zhu for their continuous support during and after my internship at Google DeepMind. Additionally, I would like to express my sincere thanks to Yangqiu Song, who has collaborated with me since my undergraduate studies and has continually influenced me during my Ph.D. Moreover, I would like to acknowledge and thank my other collaborators during my internships, including Bryan Catanzaro, Heng-Tze Cheng, Cosmo Du, Mike Dusenberry, Haoming Jiang, Zheng Li, Swaroop Mishra, Jasper Snoek, Mohammad Shoeybi, Albert Webson, Peng Xu, Jingfeng Yang, Bing Yin, Adams Wei Yu, Chao Zhang, Huaixiu Steven Zheng, Hao Zhou, and others.

Next, I would like to thank my colleagues and friends at UIUC, including Pritom Saha Akash, Nishant Balepur, Ziyi Chen, Yinfang Chen, Yihan Du, Junyi Gao, Zhe Huang, Jinyuan Jia, Bowen Jin, Yuhong Li, Yunzhe Li, Xinyu Lian, Chenzhengyi Liu, Yuxiang Liu, Hanyin Shao, Ruijie Wang, Boxin Wang, Haoxiang Wang, Chaoqi Yang, Zixuan Zhang, Shen Zheng, Kerui Zhu, and others. I also want to thank my academic collaborators and friends outside UIUC and the internships, including Jun Huang, Haoran Li, Bill Yuchen Lin, Hongbin Liu, Ruibo Liu, Xin Liu, Jianwei Wang, Yizhong Wang, Shaoan Xie, Canwen Xu, Nathan Yan, Fanghua Ye, Xiang Yue, Zining Zhu, and many others.

Last, I extend my deepest gratitude to my family for their unconditional love and support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The landscape of artificial intelligence has witnessed a significant shift, with rapid advancements ushering in the era of Large Language Models (LLMs). Characterized by their extensive parameter counts and unparalleled text generation capabilities, LLMs have shown promising results across a wide range of applications [6, 24, 37, 178, 179]. Their remarkable ability to understand, generate, and interact with humans has propelled them to the forefront of technological innovation, setting new benchmarks in various applications.

However, despite their advancements, current LLMs are not without significant shortcomings. Firstly, they are prone to generating inaccurate or fabricated information, which undermines their reliability. Secondly, they pose various ethical challenges, making their application a subject of considerable risk. While the enhancement of capabilities of LLMs is ongoing, ensuring ethical application becomes increasingly crucial to prevent misuse or harmful outcomes. In this dissertation, we present analyses to understand the capabilities and risks of LLMs and propose techniques for their better usage, with the goal of achieving high capability with minimal associated risk.

In Part I, we explore the capabilities of LLMs, specifically focusing on reasoning. Reasoning is a fundamental aspect of human cognition, enabling us to understand the world, draw inferences, make decisions, and solve problems. We begin with an overview of the current state of reasoning in LLMs, providing insights for future research (Chapter 2). Then, we specifically focus on the analysis of an important aspect of reasoning and cognition: self-correction. Our study indicates that despite the optimism surrounding this aspect, current LLMs cannot self-correct their reasoning intrinsically. This highlights their limited ability to improve themselves without external feedback (Chapter 3).

The power of LLMs also raises risks, including privacy issue, bias, and potential misuse. A primary area of concern is the risk of data leakage, especially the inadvertent release of personally identifiable information (PII). In Part II, we unpack this issue in LLMs. Specifically, we first test whether LLMs are prone to leak PII such as email address. We discover that these models do leak personal information due to the memorization of their training data. However, since the models are weak at associating email addresses with corresponding person names, the risk of specific personal information being extracted by attackers is relatively low (Chapter 4). Based on this foundation, we conduct a comprehensive analysis of the association capabilities of LLMs and its implications on privacy leakage. Despite the proportion of accurately predicted PII being relatively small, LLMs still demonstrate the capability to predict specific instances of email addresses and phone numbers when provided with appropriate prompts. These findings underscore the potential risk to PII confidentiality posed by the evolving capabilities of LLMs, especially as they continue to expand in scale and power (Chapter 5).

Based on our analysis, LLMs face both capability limitations and ethical challenges. In Part III, we introduce techniques aimed at improving these two aspects. Drawn from our analysis, a fundamental limitation of LLMs is their limited ability to operate beyond their intrinsic cognitive boundaries. To address this, the integration of external knowledge and feedback is vital. In light of this, we develop Raven, a language model augmented with external knowledge. Despite having substantially fewer parameters than some of the most advanced models, Raven demonstrates remarkable performance in knowledge-intensive tasks (Chapter 6).

Another consideration is the form of knowledge used. Common choices include raw text, which is unstructured and cost-effective, and knowledge graphs, which offer structured but often incomplete knowledge and are labor-intensive to build. Our proposal, Descriptive Knowledge Graph, aims to merge the best of both worlds. This approach combines the flexibility of raw text with the structured nature of knowledge graphs. By utilizing sentence relation descriptions in its edges to represent relationships, this graph allows for automatic construction without the need for human annotation. This knowledge representation can serve as a useful component in enhancing the functionality of LLMs, especially in knowledge reasoning (Chapter 7).

Furthermore, we argue that a multifaceted implementation considering both the capabilities and risks of LLMs is crucial. In this implementation, we identify "citation" as a key component in building responsible and accountable LLMs. Incorporating citations can enhance content transparency and verifiability, thereby addressing some of the ethical issues associated with the deployment and usage of LLMs (Chapter 8).

Through this endeavor, we aim to uncover and extend LLMs' true potentials and limitations. In doing so, our goal is to pave the way for their responsible advancement and application, ensuring that they contribute positively and safely across various domains. Ultimately, this leads to a future where artificial intelligence is not only powerful but also principled.

Part I

Understanding Capabilities of Large Language Models

CHAPTER 2: TOWARDS REASONING IN LARGE LANGUAGE MODELS

Reasoning is a fundamental aspect of human intelligence that plays a crucial role in human cognition and activities such as problem solving, decision making, and critical thinking. In recent years, Large Language Models (LLMs) have made significant progress in natural language processing, and there is observation that these models may exhibit reasoning abilities when they are sufficiently large. However, it is not yet clear to what extent LLMs are capable of reasoning. In this chapter, we provide a comprehensive overview of the current state of knowledge on reasoning in LLMs, including techniques for improving and eliciting reasoning in these models, methods and benchmarks for evaluating reasoning abilities, findings and implications of previous research in this field, and suggestions on future directions.[1]

2.1 INTRODUCTION

Reasoning is a cognitive process that involves using evidence, arguments, and logic to arrive at conclusions or make judgments. It plays a central role in many intellectual activities, such as problem solving, decision making, and critical thinking. The study of reasoning is important in fields like psychology [254], philosophy [188], and computer science [102], as it helps individuals make decisions, solve problems, and think critically.

Recently, Large Language Models (LLMs) [24, 37, 38, 178] such as ChatGPT have made significant advancements in natural language processing and related fields. It has been shown that these models exhibit emergent behaviors, including the ability to "reason", when they are large enough [257]. For example, by providing the models with "*chain of thoughts*", i.e., reasoning exemplars, or a simple prompt "*Let's think step by step*", these models are able to answer questions with explicit reasoning steps [125, 258], e.g., "all whales are mammals, all mammals have kidneys; therefore, all whales have kidneys." This has sparked considerable interest in the community since reasoning ability is a hallmark of human intelligence that is frequently considered missed in current artificial intelligence systems [19, 164, 173, 214].

However, despite the strong performance of LLMs on certain reasoning tasks, it remains unclear whether LLMs are actually reasoning and to what extent they are capable of reasoning. For example, Kojima et al. [125] claim that "LLMs are decent zero-shot reasoners (p. 1)", while Valmeekam et al. [240] conclude that "LLMs are still far from achieving acceptable performance on common planning/reasoning tasks which pose no issues for humans to do (p. 2)." This limitation is also stated by Wei et al. [258]:

[1] The material in this chapter is based on Huang and Chang [89].

"we qualify that although chain of thought emulates the thought processes of human reasoners, this does not answer whether the neural network is actually *reasoning* (p. 9)."

Therefore, in this chapter, we aim to provide a comprehensive overview and engage in an insightful discussion on the current state of knowledge on this fast-evolving topic. We initiate our exploration with a clarification of the concept of reasoning (Section 2.2). Subsequently, we turn our attention to the techniques for enhancing/eliciting reasoning in LLMs (Section 2.3), the methods and benchmarks for evaluating reasoning in LLMs (Section 2.4), and the key findings and implications in this field (Section 2.5). Finally, we reflect on and discuss the current state of the field (Section 2.6).

2.2 WHAT IS REASONING?

Reasoning is the process of thinking about something in a logical and systematic way, using evidence and past experiences to reach a conclusion or make a decision [56, 62, 165, 253, 254]. Reasoning involves making inferences, evaluating arguments, and drawing logical conclusions based on available information. Although "reasoning" is a term that is commonly used in literature and daily life, it is also an abstract concept that can refer to many things. To help the reader better understand this concept, we summarize several main categories of reasoning that are commonly recognized:

Deductive reasoning. Deductive reasoning is a type of reasoning in which a conclusion is drawn based on the truth of the premises. In deductive reasoning, the conclusion must necessarily follow from the premises, meaning that if the premises are true, the conclusion must also be true. For example:

- Premise: All mammals have kidneys.
- Premise: All whales are mammals.
- Conclusion: All whales have kidneys.

Inductive reasoning. Inductive reasoning is a type of reasoning in which a conclusion is drawn based on observations or evidence. The conclusion is likely to be true based on the available evidence, but it is not necessarily certain. For example:

- Observation: Every time we see a creature with wings, it is a bird.
- Observation: We see a creature with wings.
- Conclusion: The creature is likely to be a bird.

Abductive reasoning. Abductive reasoning is a type of reasoning in which a conclusion is drawn based on the best explanation for a given set of observations. The conclusion is the most likely explanation based on the available evidence, but it is not necessarily certain. For example:

- Observation: The car cannot start and there is a puddle of liquid under the engine.
- Conclusion: The most likely explanation is that the car has a leak in the radiator.

Other types of reasoning include *analogical reasoning*, which involves making comparisons between two or more things in order to make inferences or arrive at conclusions; *causal reasoning*, which involves identifying and understanding the causes and effects of events or phenomena; and *probabilistic reasoning*, which involves making decisions or arriving at conclusions based on the likelihood or probability of certain outcomes.

Formal Reasoning vs Informal Reasoning. *Formal reasoning* is a systematic and logical process that follows a set of rules and principles, often used in mathematics and logic. *Informal reasoning* is a less structured approach that relies on intuition, experience, and common sense to draw conclusions and solve problems, and is often used in everyday life. Formal reasoning is more structured and reliable, while informal reasoning is more adaptable and open-ended, but may also be less reliable. We refer the reader to Bronkhorst et al. [22], Galotti [62] for a detailed distinction between them.

Reasoning in Language Models. The concept of reasoning in language models has been around for some time, but there is not a clear definition of what it entails. In the literature, the term "reasoning" is often used to refer to informal reasoning, although it is not always explicitly stated that it is informal [40, 258]. Different forms of reasoning may be used depending on the task, benchmark, or method being used, e.g., deductive reasoning [40, 42, 77], inductive reasoning [172, 269] or abductive reasoning [113, 129, 262]. In this chapter, we encompass various forms of reasoning, with a particular focus on "informal deductive reasoning" in large language models since it is a widely used form in which the conclusion is guaranteed to be true as long as the premises are true.

2.3 TOWARDS REASONING IN LARGE LANGUAGE MODELS

Reasoning, particularly multi-step reasoning, is often seen as a weakness in language models and other NLP models [19, 206, 240]. Recent research has suggested that reasoning ability may emerge in language models at a certain scale, such as models with over 100 billion parameters [40, 257, 258]. Therefore, we follow Wei et al. [257] in considering reasoning as an ability that is rarely present in small-scale models like GPT-2 [205] and BERT [44], and therefore focus on techniques applicable to improving or eliciting "reasoning"[2] in LLMs such as GPT-3 [24] and PaLM [37].

2.3.1 Fully Supervised Finetuning

Before discussing reasoning in large language models, it is worth mentioning there is research working on eliciting/improving reasoning in small language models through *fully supervised finetuning* on specific datasets. For example, Rajani et al. [208] finetune a pretrained GPT model [204] to generate rationales that explain model predictions with the built CoS-E dataset, and find that models trained with explanations perform better on commonsense question answering tasks [233]. Talmor et al. [234] train RoBERTa [153] to perform reasoning/inference based on both implicit pre-trained knowledge and explicit free-text statements. Hendrycks et al. [82] finetune pretrained language models to solve competition mathematics problems by generating full step-by-step solutions, though the accuracy is relatively low. Nye et al. [176] train language models to do multi-step reasoning for program synthesis/execution by generating "scratchpads", i.e., intermediate computations, before producing the final answers. We refer the reader to Bhargava and Ng [16], Helwe et al. [80]'s survey for more studies in this line.

There are two major limitations of fully supervised finetuning. First, it requires a dataset containing explicit reasoning, which can be difficult and time-consuming to create. Additionally, the model is only trained on a specific dataset, which limits its application to a specific domain and may result in the model relying on artifacts in the training data rather than actual reasoning to make predictions.

2.3.2 Prompting & In-Context Learning

Large language models such as GPT-3 [24] have demonstrated remarkable few-shot performance across a variety of tasks through in-context learning. These models can be prompted with a question and a few $\langle$input, output$\rangle$ exemplars to potentially solve a problem through "reasoning", either implicitly or explicitly. However, research has shown that these models still fall short when it comes to tasks that require multiple steps of reasoning to solve [19, 206, 240]. This may be due to a lack of exploration into the full capabilities of these models, as recent studies have suggested.

[2] It is important to note that the term "reasoning" in this dissertation does not necessarily imply that LLMs are truly capable of reasoning or that they are able to reason in the same way that humans do. We will discuss this issue in more detail in Section 2.6.

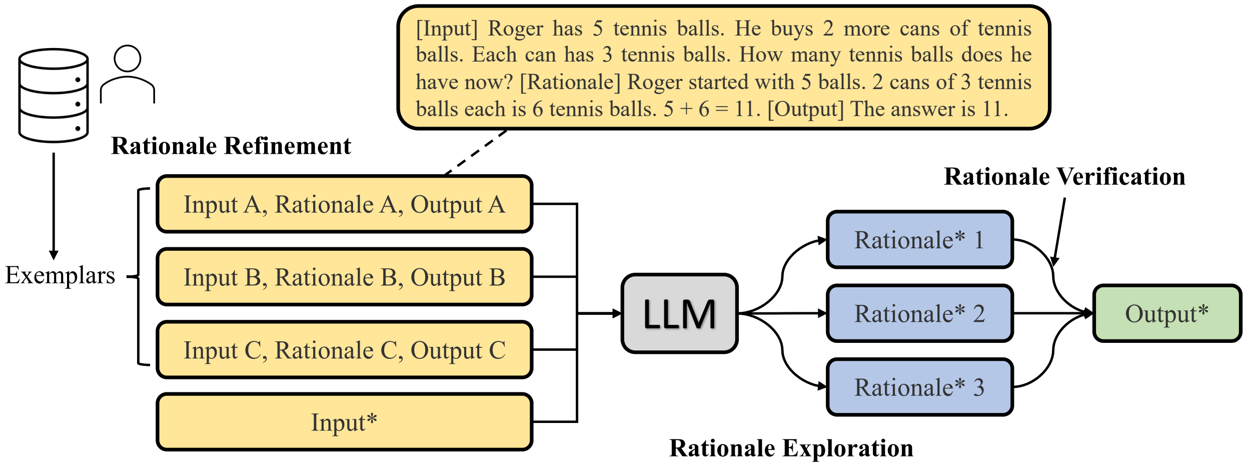


Figure 2.1: An illustration of *Chain-of-Thought Prompting* and *Rationale Engineering*, where asterisk (*) denotes the target problem to be solved.

Chain of Thought and Its Variants

To encourage LLMs to engage in reasoning rather than simply providing answers directly, we may guide LLMs to generate "reasoning" explicitly. One approach for doing this is *chain-of-thought prompting*, proposed by Wei et al. [258]. This approach involves providing a few examples of "chain of thought" (CoT), which are intermediate natural language reasoning steps, in the prompt to LLMs (Figure 2.1). Specifically, in CoT prompting, ⟨input, output⟩ demonstrations are replaced with ⟨input, *chain of thought*, output⟩ triples, e.g., "[input] 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? [*chain of thought*] Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. [output] The answer is 11." In this way, given a target question, the model learns to generate explicit rationale before producing the final answer. Experimental results show that this simple idea can improve LLMs' few-shot performance on arithmetic, symbolic, and commonsense reasoning tasks, sometimes to a striking degree.

There are several variants of chain-of-thought prompting that have been proposed in the literature, in a different form or to solve a specific problem.

Different Form: Kojima et al. [125] introduce *Zero-shot-CoT*, in which LLMs are simply prompted with the phrase "Let's think step by step" after the input, in order to elicit reasoning without the need for few-shot demonstrations. Chen et al. [32], Gao et al. [66], Madaan et al. [160] find that LLMs trained with code, e.g., Codex [30], can achieve better performance on reasoning tasks by framing reasoning as code generation. Wang et al. [246] propose to iteratively prompt chain of thought. He et al. [79] attempt to retrieve external knowledge in CoT to improve faithfulness of reasoning.

Specific Problem/Setting: Before chain of thought, Nye et al. [176] also try to use intermediate computations, named "scratchpads", to improve language models' reasoning performance in both finetuning and few-shot regimes, with a particular focus on programs. Shi et al. [221] attempt to solve multilingual reasoning tasks with CoT in the native language, CoT in English (regardless of the problem language), and CoT in English (with the problem translated to English). Chen [31] apply CoT to table-based reasoning, finding that LLMs can achieve strong performance on table tasks with only one exemplar. Prystawski et al. [201] demonstrate that CoT can improve LLMs' performance on paraphrase selection for metaphors. Lu et al. [156] apply chain of thought to solve multimodal science questions.

Rationale Engineering

The original version of chain-of-thought prompting, proposed by Wei et al. [258], relies on manually crafted examples of intermediate reasoning steps and applies greedy decoding in the generation. *Rationale engineering* aims to more effectively elicit or utilize reasoning in LLMs. This can be achieved through *rationale refinement*, which involves creating more effective examples of reasoning steps, or through *rationale exploration* and *rationale verification*, which involve exploring and verifying the rationales produced by LLMs. A summary of raltionale engineering is illustrated in Figure 2.1.

Rationale refinement. The choice of exemplars can significantly affect the few-shot performance of LLMs, as demonstrated in research such as Liu et al. [150], which also appears in chain-of-thought prompting. *Rationale refinement* aims to create and refine rationale examples that are better able to elicit reasoning in LLMs. Fu et al. [61] propose *complexity-based prompting* to create rationales with more reasoning steps. Their experiments show that the performance of LLMs improves with the increased rationale complexity. Similarly, Zhou et al. [284] propose *algorithmic prompting*, which suggests that providing more thorough examples of solutions can help improve reasoning performance on some simple math calculations. Zhang et al. [279] design *Auto-CoT* to automatically construct exemplars by partitioning questions from a given dataset into clusters and then using Zero-Shot-CoT [125] to generate the rationale for a representative question from each cluster. The analysis shows that making exemplars diverse is important in prompting LLMs to produce better rationales.

Rationale exploration. In addition to providing better exemplars, we can allow LLMs to fully explore various ways of reasoning to improve their performance on reasoning tasks, named *rationale exploration*. Based on the idea that complex problems often admit multiple ways of thinking that can lead to their unique correct answer, Wang et al. [252] present a decoding strategy called *self-consistency* to improve upon the traditional greedy decoding used in chain-of-thought prompting. This strategy involves sampling a diverse set of rationales, rather than just the greedy one, and selecting the most consistent answer by marginalizing out the sampled rationales. The idea is also used in Fu et al. [61] to vote over the top complex rationales. To further improve performance, Li et al. [142] suggest providing different demonstrations for each question by sampling exemplars from an exemplar base, in order to increase the diversity of the sampled rationales.

Rationale verification. Ensuring that the rationales produced by LLMs are valid is critical, as incorrect rationales can lead to incorrect final predictions [271]. To address this issue, the process of *rationale verification* aims to verify whether the rationales produced by LLMs lead to the correct final answers. Cobbe et al. [40] propose augmenting LLMs with a trained verifier that assigns a score to each rationale and solution generated by the LLM, selecting the highest-ranked solution as the final answer when solving math word problems. Li et al. [142] also use this technique to guide rationale selection, in conjunction with the process of rationale exploration. Different from the above methods that train an external verifier to verify the rationales, Weng et al. [261] suggest using LLMs themselves as the verifiers.

Problem Decomposition

Chain-of-thought prompting, while effective for eliciting reasoning in LLMs, can struggle with complex tasks, e.g., tasks that require compositional generalization [117, 128]. To solve a complex problem, it is helpful to first break it down into smaller, more manageable subproblems. By solving each of these subproblems, we can effectively solve the complex problem. This technique is called *problem decomposition* or *divide and conquer* [169, 194, 232].

Based on this idea, Zhou et al. [282] propose *least-to-most prompting*, which consists of two steps: decomposing the complex problem into subproblems and solving these subproblems in a specific order, with each subproblem being facilitated by the answers obtained from previously solved subproblems. As follow-up work, Drozdov et al. [48] introduce *dynamic least-to-most prompting*, which is designed to solve more realistic semantic parsing problems by decomposing the problems with prompting-based syntactic parsing and dynamically selecting exemplars based on the decomposition. In addition, Khot et al. [120] design *decomposed prompting*, which breaks down a complex problem into subproblems that can be handled by a shared library of prompting-based LLMs, each specialized in a particular subproblem. Furthermore, Dua et al. [50] develop *successive prompting*, which iteratively decomposes a complex problem into a simple problem, with the next subproblem prediction having access to the answers to the previous subproblems. While the above methods decompose or solve compositional questions with multiple forward passes, Press et al. [200] suggest decomposing and solving the input question in one forward pass using CoT prompting. Overall, these techniques show promise for helping LLMs to solve complex tasks by decomposing the problem into more manageable subproblems.

Others

There are other techniques that have been developed to facilitate reasoning in LLMs for specific tasks or settings. For instance, Creswell and Shanahan [41], Creswell et al. [42] introduce a *selection-inference* framework that uses LLMs as modules to select and infer reasoning steps from a set of facts that culminate in the final answer. Kazemi et al. [116] suggest using backward chaining, i.e., from goal to the set of facts that support it, instead of forward chaining like Creswell and Shanahan [41], Creswell et al. [42]. In addition, Jung et al. [113] propose a method for solving binary questions by prompting LLMs abductively and recursively to rationalize each option. Zhou et al. [283] design a technique for performing numerical reasoning on complex numbers by replacing the complex numbers with simple numbers to produce simpler expressions, and then using these expressions to perform calculations on the complex numbers. There are also efforts to distill reasoning from LLMs into smaller models, such as the work by Li et al. [140], Magister et al. [161], Shridhar et al. [226]. Finally, we refer the reader to Dohan et al. [46]'s position paper on *language model cascade*, which presents a unifying framework for understanding chain-of-thought prompting and research in this line.

2.3.3 Hybrid Method

While "prompting" techniques can help elicit or better utilize reasoning in large language models to solve reasoning tasks, they do not actually improve the reasoning capabilities of the LLMs themselves, as the parameters of the models remain unchanged. In contrast, the "hybrid approach" aims to simultaneously improve the reasoning capabilities of LLMs and make better use of these models in order to solve complex problems. This approach involves both enhancing the reasoning capabilities of the LLMs and using techniques such as prompting to effectively utilize these capabilities.

Reasoning-Enhanced Training and Prompting

One approach to improving the reasoning capabilities of LLMs is to pretrain or finetune the models on datasets that include "reasoning". Lewkowycz et al. [137], Taylor et al. [236] find that LLMs trained on datasets containing scientific and mathematical data can achieve better performance on reasoning tasks like quantitative reasoning problems when using CoT prompting[3]. Pi et al. [198] show that continually pretraining with SQL data can boost the performance of language models, e.g., T5 [207], on natural language reasoning such as numerical reasoning and logical reasoning. Furthermore, Chung et al. [38] develop Flan models by finetuning PaLM [37] and T5 [207] with 1.8k finetuning tasks, including CoT data, and find that CoT data are critical to keeping reasoning abilities. Similarly, Yu et al. [273] finetune OPT [277] on 10 reasoning datasets and observe that it can improve some reasoning capabilities of LLMs. Anil et al. [5] study the length generalization abilities of LLMs, i.e., whether LLMs learned with short problem instances can generalize to long ones. They discover that the combination of few-shot scratchpad (or chain of thought) finetuning and scratchpad prompting results in a significant improvement in LLMs' ability to generalize to longer problems, while this phenomenon is not observed in the standard fully supervised finetuning paradigm.

Bootstrapping & Self-Improving

Instead of finetuning LLMs on pre-built datasets that include reasoning, there are studies that have explored the idea of using LLMs to self-improve their reasoning abilities through a process known as bootstrapping. One example of this is the *Self-Taught Reasoner (STaR)* introduced by Zelikman et al. [276], in which a LLM is trained and refined on its own output iteratively. Specifically, with CoT prompting, the model first generates initial rationales. And then, the model is finetuned on rationales that lead to correct answers. This process can be repeated, with each iteration resulting in an improved model that can generate better training data, which in turn leads to further improvements. As a follow-up to this work, Huang et al. [86] show that LLMs are able to self-improve their reasoning abilities without the need for supervised data by leveraging the self-consistency of reasoning [252].

[3] This may also be true for models trained with code [30, 60].

2.4 MEASURING REASONING IN LARGE LANGUAGE MODELS

We summarize methods and benchmarks for evaluating reasoning abilities of LLMs in this section.

2.4.1 End Task Performance

One way to measure reasoning abilities of LLMs is to report their performance, e.g., accuracy, on end tasks that require reasoning. We list some common benchmarks as follows.

Arithmetic Reasoning. *Arithmetic reasoning* is the ability to understand and apply mathematical concepts and principles in order to solve problems involving arithmetic operations. This involves using logical thinking and mathematical principles to determine the correct course of action when solving mathematical problems. Representative benchmarks for arithmetic reasoning include GSM8K [40], Math [82], MathQA [4], SVAMP [191], ASDiv [168], AQuA [148], and MAWPS [211]. It is worth mentioning that Anil et al. [5] generate the *Parity Datasets* and the *Boolean Variable Assignment Dataset* for analyzing the length generalization capabilities of LLMs (Section 2.3.3).

Commonsense Reasoning. *Commonsense Reasoning* is the use of everyday knowledge and understanding to make judgments and predictions about new situations. It is a fundamental aspect of human intelligence that enables us to navigate our environment, understand others, and make decisions with incomplete information. Benchmarks that can be used for testing commonsense reasoning abilities of LLMs include CSQA [233], StrategyQA [68], and ARC [39]. We refer the reader to Bhargava and Ng [16]'s survey for more work in this domain.

Symbolic Reasoning. *Symbolic reasoning* is a form of reasoning that involves the manipulation of symbols according to formal rules. In symbolic reasoning, we use abstract symbols to represent concepts and relationships, and then manipulate those symbols according to precise rules in order to draw conclusions or solve problems. Two benchmarks of symbolic reasoning are presented in Wei et al. [258], including Last Letter Concatenation and Coin Flip.

Others. In practice, there are many benchmarks that can be used to evaluate reasoning abilities of LLMs (indirectly), as long as the downstream task involves reasoning. BIG-bench [229], for example, includes over 200 tasks that test a range of reasoning skills, including tasks like Date Understanding, Word Sorting, and Causal Judgement. Other benchmarks, such as SCAN [128] and the one proposed by Anil et al. [5], focus on evaluating generalization ability. LLMs can also be tested on their table reasoning abilities using benchmarks such as WikiTableQA [189], FetaQA [174], as suggested by Chen [31].

2.4.2 Analysis on Reasoning

Although LLMs have demonstrated impressive performance on various reasoning tasks, the extent to which their predictions are based on true reasoning or simple heuristics is not always clear. This is because most existing evaluations focus on their accuracy on end tasks, rather than directly assessing their reasoning steps. While some error analysis has been conducted on the generated rationales of LLMs [125, 258], this analysis has often been limited in depth.

There have been some efforts to develop metrics and benchmarks that enable a more formal/deep analysis of reasoning in LLMs. Golovneva et al. [69] design ROSCOE, a set of interpretable, detailed step-by-step evaluation metrics covering various perspectives including semantic alignment, logical inference, semantic similarity, and language coherence. Saparov and He [216] create a synthetic dataset called PrOntoQA that is generated from real or fictional ontologies. Each example in the dataset has a unique proof, which can be converted to simple sentences and back again, allowing for a formal analysis of each reasoning step. Han et al. [76] introduce a dataset called FOLIO to test the first-order logic reasoning capabilities of LLMs. FOLIO contains first-order logic reasoning problems that require models to determine the correctness of conclusions given a set of premises. In addition, Wang et al. [247] conduct ablation experiments on CoT and find that LLMs may also perform reasoning while prompting with invalid rationals. Their study also suggests that being relevant to the query and correctly ordering the reasoning steps are important for CoT prompting.

In summary, most existing studies primarily report the performance of the models on downstream reasoning tasks, without a detailed examination of the quality of the rationales produced. This leaves open the question of whether the models are actually able to reason in a way that is similar to human reasoning, or whether they are simply able to achieve good performance on the tasks through other means. Further research is needed to more formally analyze the reasoning abilities of LLMs.

2.5 FINDINGS AND IMPLICATIONS

In this section, we summarize the important findings and implications of studies on reasoning in large language models.

Reasoning seems an emergent ability of LLMs. Suzgun et al. [231], Wei et al. [257, 258] show that reasoning ability appears to emerge only in large language models like GPT-3 175B, as evidenced by significant improvements in performance on reasoning tasks at a certain scale (e.g., 100 billion parameters). This suggests that it may be more effective to utilize large models for general reasoning problems rather than training small models for specific tasks. However, the reason for this emergent ability is not yet fully understood. We refer the reader to Fu et al. [60], Wei et al. [257] for some potential explanations.

Chain of thought elicits "reasoning" of LLMs. The use of chain-of-thought (CoT) prompts [258] has been shown to improve the performance of LLMs on various reasoning tasks, as demonstrated in the experiments of Suzgun et al. [231], Wei et al. [257, 258]. Additionally, Saparov and He [216] (Section 2.4.2) find that, when using CoT prompts, LLMs are able to produce valid individual proof steps, even when the synthetic ontology is fictional or counterfactual. However, they may sometimes choose the wrong steps when multiple options are available, leading to incomplete or incorrect proofs. Moreover, for many reasoning tasks where the performance of standard prompting grows smoothly with model scale, chain-of-thought prompting can lead to dramatic performance improvement. In addition to these benefits, the use of CoT prompts has been shown to improve the out-of-distribution robustness of LLMs [5, 258, 282], an advantage that is not typically observed with standard prompting or fully supervised finetuning paradigms.

LLMs show human-like content effects on reasoning. According to Dasgupta et al. [43], LLMs exhibit reasoning patterns that are similar to those of humans as described in the cognitive literature. For example, the models' predictions are influenced by both prior knowledge and abstract reasoning, and their judgments of logical validity are impacted by the believability of the conclusions. These findings suggest that, although language models may not always perform well on reasoning tasks, their failures often occur in situations that are challenging for humans as well. This provides some evidence that language models may "reason" in a way that is similar to human reasoning.

LLMs are still unskilled at complex reasoning. Although LLMs seem to possess impressive reasoning capabilities with the techniques described in Section 2.3, they still struggle with more complex reasoning tasks or those involving implicature, according to studies such as Han et al. [76], Ruis et al. [213], Valmeekam et al. [240]. For instance, Valmeekam et al. [240] find that even in relatively simple commonsense planning domains that humans would have no trouble navigating, LLMs such as GPT-3 [24] and BLOOM [217] struggle to perform effectively. These findings suggest that existing benchmarks may be too simple to accurately gauge the true reasoning abilities of LLMs, and that more challenging tasks may be needed to fully evaluate their abilities in this regard.

2.6 REFLECTION, DISCUSSION, AND FUTURE DIRECTIONS

Why reasoning? Reasoning is the process of thinking about something in a logical and systematic way, and it is a key aspect of human intelligence. By incorporating reasoning capabilities into language models, we can enable them to perform tasks that require more complex and nuanced thinking, such as problem solving, decision making, and planning [100, 101, 228]. This can improve the performance of these models on downstream tasks and increase their out-of-distribution robustness [5, 231, 257, 258, 282]. In addition, reasoning can make language models more explainable and interpretable, as it provides explicit rationales for their predictions.

Right task/application? As Valmeekam et al. [240] point out, current benchmarks may not adequately reflect the reasoning capabilities of LLMs. In addition, tasks such as solving simple math problems and concatenating letters in strings (Section 2.4.1) are artificial and do not accurately reflect real-world situations. To truly understand the reasoning ability of LLMs, it is important to consider more realistic and meaningful applications such as decision making [53], legal reasoning [134], and scientific reasoning [287]. Our ultimate goal should not be to enable LLMs to solve simple math problems, which can be simply done with other programs. When conducting relevant research, it is essential to ask *whether the specific task being tackled is meaningful* and *whether the proposed method can be generalized to more realistic tasks and applications*.

Are language models really able to reason? There are several indications that LLMs are able to reason, including 1) high performance on various tasks requiring reasoning [231]; 2) the ability to reason step-by-step with chain-of-thought prompting [258]; and 3) the reflection of human-like content effects on reasoning [43]. However, these findings are not sufficient to conclude that LLMs can truly reason. For 1), it is not clear whether the models are making predictions based on *reasoning* or *heuristics* [191]. For many existing benchmarks on reasoning, actually, we can design a program with heuristic rules to achieve very high performance. We usually do not think a program relying on heuristic rules is capable of reasoning. For 2), although the models seem to reason step-by-step, the generated rationales may be incorrect and inconsistent. It is possible that the models are "generating reasoning-like response" rather than "reasoning step-by-step". For 3), while LLMs display some human-like reasoning patterns, this does not necessarily mean that they behave like humans.

Additionally, there are several observations that suggest LLMs may not be capable of reasoning: 1) LLMs still struggle with tasks that require complex reasoning [76, 213, 240]. If LLMs are really decent reasoners, they should handle tasks that can be simply solved by humans through reasoning; 2) LLMs make mistakes in their reasoning, as explained above; 3) The performance of LLMs on downstream tasks has been found to be sensitive to the frequency of certain terms, such as numbers, in the training data [113, 210], which would not be expected if the models were solving mathematical problems through reasoning; 4) Language models have been found to struggle with associating relevant information that they have memorized [95]; 5) Current LLMs cannot self-correct their reasoning intrinsically [92].

Overall, it is still too early to draw a conclusion about the proposed question. In fact, there is also an ongoing debate about whether language models can actually *understand* language or capture *meaning* [15, 138, 163, 199]. Further in-depth analysis of factors such as training data, model architecture, and optimization objectives is needed, as well as the development of better benchmarks for measuring the reasoning capabilities of LLMs. However, it is clear that the current models are not yet capable of robust reasoning.

Improving reasoning capabilities of LLMs. While techniques like chain-of-thought prompting [258] may help to elicit reasoning abilities in large language models, they cannot enable the models to solve tasks beyond their current capabilities. To truly enhance reasoning in LLMs, we need to utilize training data, model architecture, and optimization objectives that are designed to encourage reasoning. For example, finetuning a model with a dataset including CoT data has been shown to improve reasoning [38], and models can also self-improve through the process of bootstrapping their reasoning [86, 276]. There is still much research that needs to be done in this area, and we look forward to future progress in improving reasoning in large language models.

2.7 CONCLUSION

In this chapter, we have provided a detailed and up-to-date review of the current state of knowledge on reasoning in large language models. We have discussed techniques for improving and eliciting reasoning in LLMs, methods and benchmarks for evaluating reasoning abilities, and the findings and implications of previous studies in this topic. While LLMs have made significant progress in natural language processing and related fields, it remains unclear to what extent they are capable of true reasoning or whether they are simply using memorized patterns and heuristics to solve problems. Further research is needed to fully understand the reasoning abilities of LLMs, improve LLMs' reasoning capabilities, and determine their potential for use in a variety of applications.

CHAPTER 3: LARGE LANGUAGE MODELS CANNOT SELF-CORRECT REASONING YET

Large Language Models (LLMs) have emerged as a groundbreaking technology with their unparalleled text generation capabilities across various applications. Nevertheless, concerns persist regarding the accuracy and appropriateness of their generated content. A contemporary methodology, *self-correction*, has been proposed as a remedy to these issues. Building upon this premise, in this chapter, we critically examine the role and efficacy of self-correction within LLMs, shedding light on its true potential and limitations. Central to our investigation is the notion of *intrinsic self-correction*, whereby an LLM attempts to correct its initial responses based solely on its inherent capabilities, without the crutch of external feedback. In the context of reasoning, our research indicates that LLMs struggle to self-correct their responses without external feedback, and at times, their performance even degrades after self-correction. Drawing from these insights, we offer suggestions for future research and practical applications in this field.[4]

3.1 INTRODUCTION

The rapid advancements in the domain of artificial intelligence have ushered in the era of Large Language Models (LLMs). These models, characterized by their expansive parameter counts and unparalleled capabilities in text generation, have showcased promising results across a multitude of applications [7, 37, 179]. However, concerns about their accuracy, reasoning capabilities, and the safety of their generated content have drawn significant attention from the community [3, 13, 27, 95, 139, 219, 220, 255, 259, 280, 282, 288].

Amidst this backdrop, the concept of "self-correction" has emerged as a promising solution, where LLMs refine their responses based on feedback to their previous outputs [11, 34, 63, 65, 121, 159, 182, 193, 225, 260]. However, the underlying mechanics and efficacy of self-correction in LLMs remain underexplored. A fundamental question arises: If an LLM possesses the ability to self-correct, why doesn't it simply offer the correct answer in its initial attempt? This chapter delves deeply into this paradox, critically examining the self-correction capabilities of LLMs, with a particular emphasis on reasoning [89, 258, 282].

To study this, we first define the concept of *intrinsic self-correction*, a scenario wherein the model endeavors to rectify its initial responses based solely on its inherent capabilities, without the crutch of external feedback. Such a setting is crucial because high-quality external feedback is often unavailable in many real-world applications. Moreover, it is vital

[4]The material in this chapter is based on Huang et al. [92].

to understand the intrinsic capabilities of LLMs. Contrary to the optimism surrounding self-correction [121, 159, 182, 225], our findings indicate that LLMs struggle to self-correct their reasoning in this setting. In most instances, the performance after self-correction even deteriorates. This observation is in contrast to prior research such as Kim et al. [121], Shinn et al. [225]. Upon closer examination, we observe that the improvements in these studies result from using oracle labels to guide the self-correction process, and the improvements vanish when oracle labels are not available.

Besides the reliance on oracle labels, we also identify other issues in the literature regarding measuring the improvement achieved by self-correction. First, we note that self-correction, by design, utilizes multiple LLM responses, thus making it crucial to compare it to baselines with equivalent inference costs. From this perspective, we investigate multi-agent debate [49, 143] as a means to improve reasoning, where multiple LLM instances (can be multiple copies of the same LLM) critique each other's responses. However, our results reveal that its efficacy is no better than self-consistency [252] when considering an equivalent number of responses, highlighting the limitations of such an approach.

Another important consideration for self-correction involves prompt design. Specifically, each self-correction process involves designing prompts for both the initial response generation and the self-correction steps. Our evaluation reveals that the self-correction improvement claimed by some existing work stems from the sub-optimal prompt for generating initial responses, where self-correction corrects these responses with more informative instructions about the initial task in the feedback prompt. In such cases, simply integrating the feedback into the initial instruction can yield better results, and self-correction again decreases performance.

In light of our findings, we provide insights into the nuances of LLMs' self-correction capabilities and initiate discussions to encourage future research focused on exploring methods that can genuinely correct reasoning.

3.2 BACKGROUND AND RELATED WORK

With the LLM evolution, the notion of self-correction gained prominence. The discourse on self-correction pivots around whether these advanced models can recognize the correctness of their outputs and provide refined answers [11, 159, 260]. For example, in the context of mathematical reasoning, an LLM might initially solve a complex problem but make an error in one of the calculation steps. In an ideal self-correction scenario, the model is expected to recognize the potential mistake, revisit the problem, correct the error, and consequently produce a more accurate solution.

Method	Issue
RCI [121]; Reflexion [225]	Use of oracle labels (Section 3.3)
Multi-Agent Debate [49]	Unfair comparison (Section 3.4)
Self-Refine [159]	Sub-optimal prompt design (Section 3.5)

Table 3.1: Summary of issues in previous LLM self-correction evaluation.

Yet, the definition of "self-correction" varies across the literature, leading to ambiguity. A pivotal distinction lies in the source of feedback [182], which can purely come from the LLM, or can be drawn from external inputs. Internal feedback relies on the model's inherent knowledge and parameters to reassess its outputs. In contrast, external feedback incorporates inputs from humans, other models [193, 251], or external tools and knowledge sources [34, 65, 70, 177].

In this work, we focus on examining the self-correction capability of LLMs for reasoning. Reasoning is a fundamental aspect of human cognition, enabling us to understand the world, draw inferences, make decisions, and solve problems. To enhance the reasoning performance of LLMs, Kim et al. [121], Shinn et al. [225] use oracle labels about the answer correctness to guide the self-correction process. However, in practice, high-quality external feedback such as answer correctness is often unavailable. For effective self-correction, the ability to judge the correctness of an answer is crucial and should ideally be performed by the LLM itself. Consequently, our focus shifts to self-correction without any external or human feedback. We term this setting **intrinsic self-correction**. For brevity, unless explicitly stated otherwise (e.g., self-correction with oracle feedback), all references to "self-correction" in the remainder of this chapter pertain to intrinsic self-correction.

In the following sections, we will evaluate a variety of existing self-correction techniques. We demonstrate that existing techniques actually decrease reasoning performance when oracle labels are not used (Section 3.3), perform worse than methods without self-correction when utilizing the same number of model responses (Section 3.4), and lead to less effective outcomes when using informative prompts for generating initial responses (Section 3.5). We present an overview of issues in the evaluation setups of previous LLM self-correction works in Table 3.1, with detailed discussions in the corresponding sections.

3.3 LLMS CANNOT SELF-CORRECT REASONING INTRINSICALLY

In this section, we evaluate existing self-correction methods and compare their performance with and without oracle labels regarding the answer correctness.

3.3.1 Experimental Setup

Benchmarks. We use datasets where existing self-correction methods with oracle labels have demonstrated significant performance improvement, including

- **GSM8K** [40]: GSM8K comprises a test set of 1,319 linguistically diverse grade school math word problems, curated by human problem writers. There is a notable improvement of approximately 7% as evidenced by Kim et al. [121] after self-correction.
- **CommonSenseQA** [233]: This dataset offers a collection of multi-choice questions that test commonsense reasoning. An impressive increase of around 15% is showcased through the self-correction process, as demonstrated by Kim et al. [121]. Following Kim et al. [121], Kojima et al. [125], we utilize the dev set for our evaluation, which encompasses 1,221 questions.
- **HotpotQA** [268]: HotpotQA is an open-domain multi-hop question answering dataset. Shinn et al. [225] demonstrate significant performance improvement through self-correction. We test models' performance in a closed-book setting and evaluate them using the same set as Shinn et al. [225]. This set contains 100 questions, with exact match serving as the evaluation metric.

Test Models and Setup. We first follow Kim et al. [121], Shinn et al. [225] to evaluate the performance of self-correction with oracle labels, using GPT-3.5-Turbo (`gpt-3.5-turbo-0613`) and GPT-4 accessed on 2023/08/29. For intrinsic self-correction, to provide a more thorough analysis, we also evaluate GPT-4-Turbo (`gpt-4-1106-preview`) and Llama-2 (`Llama-2-70b-chat`) [238]. For GPT-3.5-Turbo, we employ the full evaluation set. For other models, to reduce the cost, we randomly sample 200 questions for each dataset (100 for HotpotQA) for testing. We prompt the models to undergo a maximum of two rounds of self-correction. We use a temperature of 1 for GPT-3.5-Turbo and GPT-4, and a temperature of 0 for GPT-4-Turbo and Llama-2, to provide evaluation across different decoding algorithms.

Prompts. Following Kim et al. [121], Shinn et al. [225], we apply a three-step prompting strategy for self-correction: 1) prompt the model to perform an initial generation (which also serves as the results for Standard Prompting); 2) prompt the model to review its previous generation and produce feedback; 3) prompt the model to answer the original question again with the feedback.

For our experiments, we mostly adhere to the prompts from the source papers. For GSM8K and CommonSenseQA, we integrate format instructions into the prompts of Kim et al. [121] to facilitate a more precise automatic evaluation (detailed prompts can be found

		GSM8K	CommonSenseQA	HotpotQA
GPT-3.5	Standard Prompting	75.9	75.8	26.0
	Self-Correct (Oracle)	84.3	89.7	29.0
GPT-4	Standard Prompting	95.5	82.0	49.0
	Self-Correct (Oracle)	97.5	85.5	59.0

Table 3.2: Results of GPT-3.5 and GPT-4 on reasoning benchmarks with oracle labels.

in Appendix A.1). For HotpotQA, we use the same prompt as Shinn et al. [225]. We also assess the performance of various self-correction prompts for intrinsic self-correction. For example, we use "*Assume that this answer could be either correct or incorrect. Review the answer carefully and report any serious problems you find.*" as the default feedback prompt for the evaluation on GPT-4-Turbo and Llama-2.

3.3.2 Results

Self-Correction with Oracle Labels. Following previous works [121, 225], we use the correct label to determine when to stop the self-correction loop. This means we utilize the ground-truth label to verify whether each step's generated answer is correct. If the answer is already correct, no (further) self-correction will be performed. Table 3.2 summarizes the results of self-correction under this setting, showcasing significant performance improvements, consistent with the findings presented in Kim et al. [121], Shinn et al. [225].

However, these results require careful consideration. For reasoning tasks, like solving mathematical problems, the availability of oracle labels seems counter-intuitive. If we are already in possession of the ground truth, there seems to be little reason to deploy LLMs for problem-solving. Therefore, the results can only be regarded as indicative of an oracle's performance.

Intrinsic Self-Correction. Per the above discussion, performance improvements achieved using oracle labels do not necessarily reflect true self-correction ability. Therefore, we turn our focus to the results in the *intrinsic self-correction* setting as defined in Section 3.2. To achieve this, we eliminate the use of labels, requiring LLMs to independently determine when to stop the self-correction process, i.e., whether to retain their previous answers.

Tables 3.3 and 3.4 report the accuracies and the number of model calls. We observe that, after self-correction, the accuracies of all models drop across all benchmarks.

To provide a more comprehensive assessment, we also design several different self-correction prompts to determine if there are better prompts that could enhance reasoning

		# calls	GSM8K	CommonSenseQA	HotpotQA
GPT-3.5	Standard Prompting	1	**75.9**	**75.8**	**26.0**
	Self-Correct (round 1)	3	75.1	38.1	25.0
	Self-Correct (round 2)	5	74.7	41.8	25.0
GPT-4	Standard Prompting	1	**95.5**	**82.0**	**49.0**
	Self-Correct (round 1)	3	91.5	79.5	**49.0**
	Self-Correct (round 2)	5	89.0	80.0	43.0

Table 3.3: Results of GPT-3.5 and GPT-4 on reasoning benchmarks with intrinsic self-correction.

		# calls	GSM8K	CommonSenseQA
GPT-4-Turbo	Standard Prompting	1	**91.5**	**84.0**
	Self-Correct (round 1)	3	88.0	81.5
	Self-Correct (round 2)	5	90.0	83.0
Llama-2	Standard Prompting	1	**62.0**	**64.0**
	Self-Correct (round 1)	3	43.5	37.5
	Self-Correct (round 2)	5	36.5	36.5

Table 3.4: Results of GPT-4-Turbo and Llama-2 with intrinsic self-correction.

performance. Nonetheless, as shown in Tables 3.5 and 3.6, without the use of oracle labels, self-correction consistently results in a decrease in performance.

3.3.3 Why does the performance not increase, but instead decrease?

Empirical Analysis. Figure 3.1 summarizes the results of changes in answers after two rounds of self-correction, with two examples of GPT-3.5 illustrated in Figure 3.2. For GSM8K, 74.7% of the time, GPT-3.5 retains its initial answer. Among the remaining instances, the model is more likely to modify a correct answer to an incorrect one than to revise an incorrect answer to a correct one. ***The fundamental issue is that LLMs cannot properly judge the correctness of their reasoning.*** For CommonSenseQA, there is a higher chance that GPT-3.5 alters its answer. The primary reason for this is that false answer options in CommonSenseQA often appear somewhat relevant to the question, and using the self-correction prompt might bias the model to choose another option, leading to a high "correct $\Rightarrow$ incorrect" ratio. Similarly, Llama-2 also frequently converts a correct answer into an incorrect one. Compared to GPT-3.5 and Llama-2, both GPT-4 and GPT-4-Turbo are more likely to retain their initial answers. This may be because GPT-4

	# calls	GSM8K	CommonSenseQA
Standard Prompting	1	91.5	84.0
Feedback Prompt: Assume that this answer could be either correct or incorrect. Review the answer carefully and report any serious problems you find.			
Self-Correct (round 1)	3	88.0	81.5
Self-Correct (round 2)	5	90.0	83.0
Feedback Prompt: Review your previous answer and determine whether it's correct. If wrong, find the problems with your answer.			
Self-Correct (round 1)	3	90.0	74.5
Self-Correct (round 2)	5	90.0	81.0
Feedback Prompt: Verify whether your answer is correct, and provide an explanation.			
Self-Correct (round 1)	3	91.0	81.5
Self-Correct (round 2)	5	91.0	83.5

Table 3.5: Results of GPT-4-Turbo with different feedback prompts.

	# calls	GSM8K	CommonSenseQA
Standard Prompting	1	62.0	64.0
Feedback Prompt: Assume that this answer could be either correct or incorrect. Review the answer carefully and report any serious problems you find.			
Self-Correct (round 1)	3	43.5	37.5
Self-Correct (round 2)	5	36.5	36.5
Feedback Prompt: Review your previous answer and determine whether it's correct. If wrong, find the problems with your answer.			
Self-Correct (round 1)	3	46.5	26.0
Self-Correct (round 2)	5	30.5	37.0
Feedback Prompt: Verify whether your answer is correct, and provide an explanation.			
Self-Correct (round 1)	3	58.0	24.0
Self-Correct (round 2)	5	41.5	43.0

Table 3.6: Results of Llama-2 with different feedback prompts.

and GPT-4-Turbo have higher confidence in their initial answers, or because they are more robust and thus less prone to being biased by the self-correction prompt.[5]

Let's take another look at the results presented in Table 3.2. These results use ground-

[5]We omit the analysis on HotpotQA because the sample size used in the source paper is quite small, which may not produce meaningful statistics.

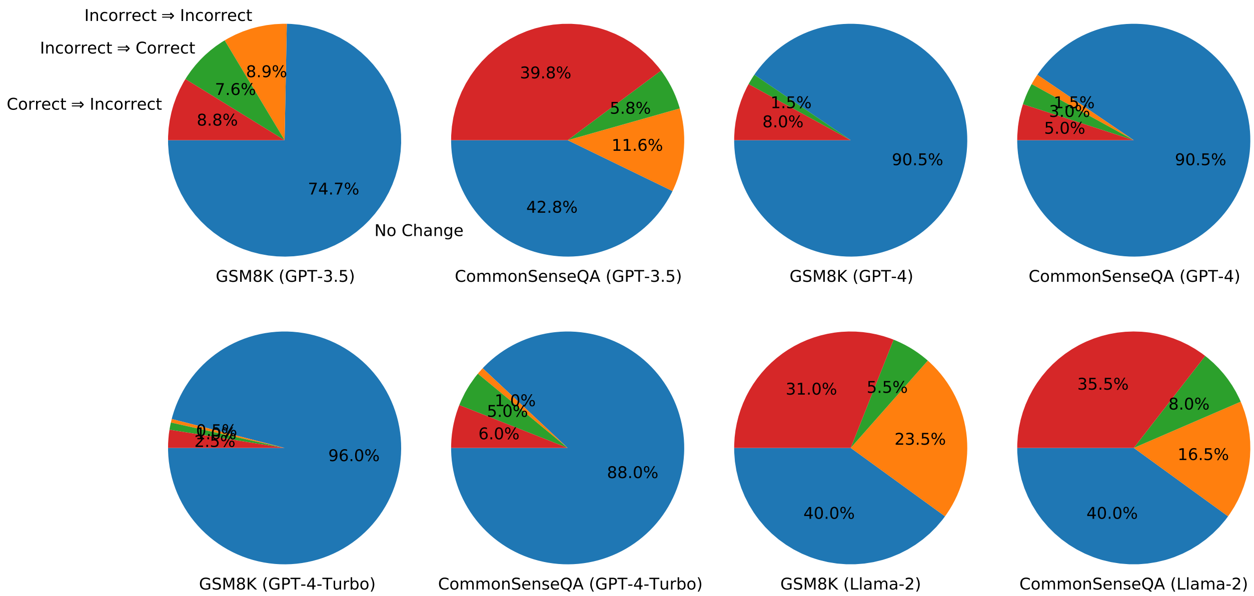


Figure 3.1: Analysis of the changes in answers after two rounds of self-correction. *No Change*: The answer remains unchanged; *Correct ⇒ Incorrect*: A correct answer is changed to an incorrect one; *Incorrect ⇒ Correct*: An incorrect answer is revised to a correct one; *Incorrect ⇒ Incorrect*: An incorrect answer is altered but remains incorrect.

truth labels to prevent the model from altering a correct answer to an incorrect one. However, ***determining how to prevent such mischanges is, in fact, the key to ensuring the success of self-correction.***

Intuitive Explanation. If the model is well-aligned and paired with a thoughtfully designed initial prompt, the initial response should already be optimal relative to the prompt and the specific decoding algorithm. Introducing feedback can be viewed as adding an additional prompt, potentially skewing the model towards generating a response that is tailored to this combined input. In an intrinsic self-correction setting, on the reasoning tasks, this supplementary prompt may not offer any extra advantage for answering the question. In fact, it might even bias the model away from producing an optimal response to the initial prompt, resulting in a performance drop.

3.4 MULTI-AGENT DEBATE DOES NOT OUTPERFORM SELF-CONSISTENCY

Another potential approach for LLMs to self-correct their reasoning involves allowing the models to critique and debate through multiple model calls [29, 49, 143]. Du et al. [49] implement a multi-agent debate method by leveraging multiple instances of a single ChatGPT model and demonstrate significant improvements on reasoning tasks. We adopt their

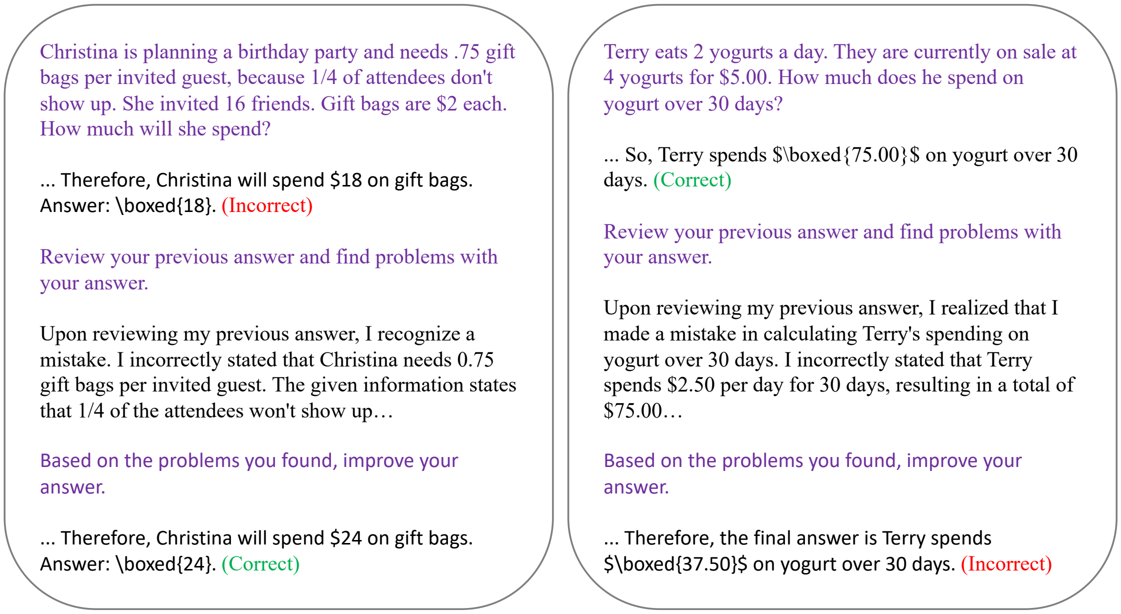


Figure 3.2: Examples on GSM8K with GPT-3.5. *Left*: successful self-correction; *Right*: failed self-correction. Full prompts and responses can be viewed in Figures A.1 and A.2 of Appendix A.1.

	# responses	GSM8K
Standard Prompting	1	76.7
Self-Consistency	3	82.5
Multi-Agent Debate (round 1)	6	83.2
Self-Consistency	6	85.3
Multi-Agent Debate (round 2)	9	83.0
Self-Consistency	9	**88.2**

Table 3.7: Results of multi-agent debate and self-consistency.

method to test performance on GSM8K. For an unbiased implementation, we use the exact same prompt as Du et al. [49] and replicate their experiment with the `gpt-3.5-turbo-0301` model, incorporating 3 agents and 2 rounds of debate. The only distinction is that, to reduce result variance, we test on the complete test set of GSM8K, compared to their usage of 100 examples. For reference, we also report the results of self-consistency [252], which prompts models to generate multiple responses and performs majority voting to select the final answer.

Table 3.7 presents the results. The results indicate that both multi-agent debate and self-consistency achieve significant improvements over standard prompting. However, when comparing multi-agent debate to self-consistency, we observe that the performance of multi-

	# calls	CommonGen-Hard
Standard Prompting*	1	44.0*
Self-Correct*	7	67.0*
Standard Prompting*	1	53.0
Self-Correct*	7	61.1
Standard Prompting (ours)	1	**81.8**
Self-Correct*	7	75.1

* Prompts and results from Madaan et al. [159].

Table 3.8: Results of Constrained Generation.

agent is only slightly better than that of self-consistency with the same number of agents (3 responses, the baseline also compared in Du et al. [49]). Furthermore, for self-consistency with an equivalent number of responses, multi-agent debate significantly underperforms simple self-consistency using majority voting.

In fact, rather than labeling the multi-agent debate as a form of "debate" or "critique", it is more appropriate to perceive it as a means to achieve "consistency" across multiple model generations. Fundamentally, its concept mirrors that of self-consistency; the distinction lies in the voting mechanism, whether voting is model-driven or purely based on counts. The observed improvement is evidently not attributed to "self-correction", but rather to "self-consistency". If we aim to argue that LLMs can self-correct reasoning through multi-agent debate, it is preferable to exclude the effects of selection among multiple generations.

3.5 PROMPT DESIGN IMPACTS SELF-CORRECTION EVALUATION

In Section 3.3, we observe that although self-correction decreases reasoning performance with all types of feedback prompts we have evaluated, performance varies with different feedback prompts. In this section, we further emphasize the importance of proper prompt design in generating initial LLM responses to fairly measure the performance improvement achieved by self-correction. For example, if a task requires that the model response should meet criteria that can be easily specified in the initial instruction (e.g., the output should contain certain words, the generated code should be efficient, the sentiment should be positive, etc.), instead of including such requirements only in the feedback prompt, an appropriate comparison would be to directly and explicitly incorporate these requirements into the prompt for generating initial responses. Otherwise, when the instruction for generating initial predictions is not informative enough, even if the performance improves, it is unclear whether the improvement merely comes from more detailed instructions in the feedback prompt or from the self-correction step itself.

To illustrate such prompt design issues in the self-correction evaluation of some prior work, we take the Constrained Generation task in Madaan et al. [159] as an example, where the task requires models to generate coherent sentences using all 20-30 input concepts. The original prompt in Madaan et al. [159] (Figure A.5) does not clearly specify that the LLM needs to include *all* concepts in the prompt; thus, they show that self-correction improves task performance by asking the model to identify missing concepts and then guiding it to incorporate these concepts through feedback.

Based on this observation, we add the following instruction "*Write a reasonable paragraph that includes *ALL* of the above concepts*" to the prompt for initial response generation (refer to Figure A.6 for the full prompt). Following Madaan et al. [159], we use concept coverage as the metric. We reference their results and replicate their experiments using `gpt-3.5-turbo-0613`. Table 3.8 demonstrates that our new prompt, denoted as *Standard Prompting (ours)*, significantly outperforms the results after self-correction of Madaan et al. [159], and applying their self-correction prompt on top of model responses from our stronger version of the standard prompting again leads to a decrease in performance.

Here, we present a simple example to illustrate how prompt design impacts self-correction performance. The effects of prompt design are complex and may also be reflected in the techniques discussed in previous sections. For instance, in the self-correction methods discussed in Section 3.3, self-correction can be performed either within the same conversation or by initiating a new one. In multi-agent debate described in Section 3.4, different personas might be assigned to different agents. There is always performance variance across different prompts. Thus, it is plausible that we could identify a self-correction prompt that enhances model performance on specific benchmarks. However, this no longer aligns with the *intrinsic self-correction* setting discussed in our study. Such a search essentially leverages feedback from humans or training examples. Moreover, the same strategy can also be effectively applied to optimize initial prompts [267, 285], possibly achieving better performance without necessitating additional model calls for self-correction. In brief, our focus is not on the question, "Is there a self-correction prompt that can enhance performance on specific benchmarks?" Such a query is not particularly meaningful. Instead, we should tackle a more foundational issue: "Are large language models really able to self-correct their reasoning based solely on their inherent capabilities?"

3.6 CONCLUSION AND DISCUSSION

Our work shows that current LLMs struggle to self-correct their reasoning without external feedback. This implies that expecting these models to inherently recognize and rectify their reasoning mistakes is overly optimistic so far. In light of these findings, it is imperative for the community to approach the concept of self-correction with a discerning perspective, acknowledging its potential and recognizing its boundaries. By doing so, we can better equip the self-correction technique to address the limitations of LLMs and develop the next generation of LLMs with enhanced capabilities. In the following, we provide insights into scenarios where self-correction shows the potential strengths and offer guidelines on the experimental design of future self-correction techniques to ensure a fair comparison.

Leveraging external feedback for correction. In this work, we demonstrate that current LLMs cannot improve their reasoning performance through intrinsic self-correction. Therefore, when valid external feedback is available, it is beneficial to leverage it properly to enhance model performance. For example, Chen et al. [34] show that LLMs can significantly improve their code generation performance through self-debugging by including code execution results in the feedback prompt to fix issues in the predicted code. In particular, when the problem description clearly specifies the intended code execution behavior, e.g., with unit tests, the code executor serves as the perfect verifier to judge the correctness of predicted programs, while the error messages also provide informative feedback that guides the LLMs to improve their responses. Gou et al. [70] demonstrate that LLMs can more effectively verify and correct their responses when interacting with various external tools such as search engines and calculators. Cobbe et al. [40], Lightman et al. [144], Wang et al. [251] train a verifier or a critique model on a high-quality dataset to verify or refine LLM outputs, which can be used to provide feedback for correcting prediction errors. Besides automatically generated external feedback, we also often provide feedback ourselves when interacting with LLMs, guiding them to produce the content we desire. Designing techniques that enable LLMs to interact with the external environment and learn from different kinds of available feedback is a promising direction for future work.

Evaluating self-correction against baselines with comparable inference costs. By design, self-correction requires additional LLM calls, thereby increasing the costs for encoding and generating extra tokens. Section 3.4 demonstrates that the performance of asking the LLM to produce a final response based on multiple previous responses, such as with the multi-agent debate approach, is inferior to that of self-consistency [252] with the same number of responses. Regarding this, we encourage future work proposing new self-correction methods to always include an in-depth inference cost analysis to substantiate claims of performance improvement. Moreover, strong baselines that leverage multiple model responses, like self-consistency, should be used for comparison. An implication for future work is to develop models with a higher probability of decoding the optimal solution in their answer distributions, possibly through some alignment techniques. This would enable the model to generate better responses without necessitating multiple generations.

Putting equal efforts into prompt design. As discussed in Section 3.5, to gain a better understanding of the improvements achieved by self-correction, it is important to include a complete task description in the prompt for generating initial responses, rather than leaving part of the task description for the feedback prompt. Broadly speaking, equal effort should be invested in designing the prompts for initial response generation and for self-correction; otherwise, the results could be misleading.

3.7 LIMITATIONS AND BROADER IMPACT

Although we have conducted a comprehensive evaluation spanning a variety of self-correction strategies, prompts, and benchmarks, our work focuses on evaluating reasoning of LLMs. Thus, it is plausible that there exist self-correction strategies that could enhance LLM performance in other domains. For example, prior works have demonstrated the successful usage of self-correction that aligns model responses with specific preferences, such as altering the style of responses or enhancing their safety [11, 63, 159]. A key distinction arises in the capability of LLMs to accurately assess their responses in relation to the given tasks. For example, LLMs can properly evaluate whether a response is inappropriate [63], but they may struggle to identify errors in their reasoning.

Furthermore, several prior works have already shown that LLM self-correction performance becomes significantly weaker without access to external feedback [70, 281] and can be easily biased by misleading feedback [248], which is consistent with our findings in this work. However, we still identified prevailing ambiguity in the wider community. Some existing literature may inadvertently contribute to this confusion, either by relegating crucial details about label usage to less prominent sections or by failing to clarify that their designed self-correction strategies actually incorporate external feedback. Regarding this, our work serves as a call to action, urging researchers to approach this domain with a discerning and critical perspective. We also encourage future research to explore approaches that can genuinely enhance reasoning.

Part II

Understanding Risks of Large Language Models

CHAPTER 4: ARE LARGE PRE-TRAINED LANGUAGE MODELS LEAKING YOUR PERSONAL INFORMATION?

Large Language Models (LLMs) possess strong capabilities, but these capabilities can also bring risks. In this chapter, we focus on the privacy risks of LLMs, exploring the pressing question: *Are Large Pre-Trained Language Models Leaking Your Personal Information?* Specifically, we analyze whether pre-trained language models are prone to leaking personal information. For our analysis, we query language models for email addresses with contexts of the email address or prompts containing the owner's name. We find that language models do leak personal information due to *memorization*. However, since the models are weak at *association*, the risk of specific personal information being extracted by attackers is relatively low.[6]

4.1 INTRODUCTION

Large Language Models (LLMs) [24, 44, 203] have taken a significant leap in a wide range of NLP tasks, attributing to the explosive growth of parameters and training data. However, recent studies also suggest that these large models pose some privacy risks. For instance, an adversary is able to recover training examples containing an individual person's name, email address, and phone number by querying the model [27]. This may lead to privacy leakage if the model is trained on a private corpus, in which case we want to improve the performance with the data [99]. Even if the data is public, LLMs may change the intended use, e.g., for information that we share but do not expect to be disseminated.

Carlini et al. [26, 27] demonstrate that LLMs memorize a lot of training data, so they are prone to leaking privacy. However, if the memorized information cannot be effectively extracted, it is still difficult for the attacker to carry out effective attacks. For instance, Lehman et al. [133] attempt to recover specific patient names and conditions with which they are associated from a BERT model that is pre-trained over clinical notes. However, they find that with their methods, the model cannot meaningfully associate names with conditions, which suggests that LLMs may not be prone to leaking personal information.

Based on existing research, we are not sure whether LLMs are safe enough in terms of preserving personal privacy. Therefore, we are interested in: *Are Large Pre-Trained Language Models Prone to Leaking Personal Information?*

To answer the above question, we first identify two capacities that may cause privacy

[6] The material in this chapter is based on Huang et al. [95]. Code and data are available at `https://github.com/jeffhj/LM_PersonalInfoLeak`.

leakage: *memorization*, i.e., LLMs memorize the personal information, thus the information can be recovered with a specific prefix, e.g., tokens before the information in the training data; and *association*, i.e., LLMs can associate the personal information with its owner, thus attackers can query the information with the owner's name, e.g., *the email address of Tom is* ___. If a model can only memorize but not associate, though the sensitive information may be leaked in some randomly generated text as shown in Carlini et al. [27], attackers cannot effectively extract specific personal information since it is difficult to find the prefix to extract the information. As far as we know, this study is the first to make this important distinction.

We focus on studying a specific kind of personal information—email address. Emails are an indispensable medium for personal/business communication. However, there are abiding problems of email fraud and spam, and the source of these problems is the leakage of personal information including email addresses.

From our experiments, we find that LLMs do leak personal information in some situations since they memorize a lot of personal information. However, the risk of a specific person's information being extracted by an interesting attacker is low since LLMs are weak at associating personal information with the information owner. We also find that some conditions, e.g., longer text patterns associated with email addresses, more knowledge about the owner, and larger scale of the model, may increase the attack success rate. Our conclusion is that LLMs like GPT-Neo [18] are relatively safe in terms of preserving personal information, but we still cannot ignore the potential privacy risks of LLMs.

4.2 RELATED WORK

Knowledge Retrieval from Language Models. Previous works have shown that large LLMs contain a significant amount of knowledge, which can be recovered by querying LLMs with appropriate prompts [21, 110, 111, 197, 250]. In this work, we attempt to extract personal information from LLMs, which can be treated as a special kind of knowledge. But unlike previous work that wants LLMs to contains as much knowledge as possible, we prefer the model to include as little personal information as possible to avoid privacy leakage.

Memorization and Privacy Risks of Language Models. Recent works have demonstrated that LLMs memorize large portions of the training data [26, 27, 237]. This may cause some privacy issues since sensitive information may be memorized in the parameters of LLMs and be leaked in some situations. Pan et al. [183] find the text embeddings from language models capture sensitive information from the plain text. Lehman et al. [133], Vakili and

Dalianis [239] study the privacy risk of sharing parameters of BERT pre-trained on clinical notes. To mitigate privacy leakage, there is a growing interest in making LLMs privacy-preserving [8, 23, 85, 141, 224, 272] by training LLMs with differential privacy guarantees [51, 52] or removing sensitive information from the training corpus.

4.3 PROBLEM STATEMENT

Our task is to measure the risk of LLMs in terms of leaking personal information. We identify two capacities of LLMs that may cause privacy leakage: *memorization* and *association*, defined as

Definition 4.1. *(Memorization) Personal information x is memorized by a model f if there exists a sequence p in the training data for f, that can prompt f to produce x using greedy decoding.*[7]

Definition 4.2. *(Association) Personal information x can be associated by a model f if there exists a prompt p (usually containing the information owner's name) designed by the attacker (who does not have access to the training data) that can prompt f to produce x using greedy decoding.*

To quantify *memorization*, an effective approach is to query the model with the context of the target sequence [26]. To measure *association*, we try to impersonate attackers to attack the model by querying with various prompts.

We focus on testing the models on email addresses. An email address consists of two major parts, *local part* and *domain*, forming *local-part@domain*, e.g., *abcf@xyz.com*. We define attack tasks based on memorization and association: 1) given the context of an email address, examine whether the model can recover the email address; 2) given the owner's name, query LLMs for the associated email address with an appropriate prompt.

4.4 DATA AND PRE-TRAINED MODEL

We test on the GPT-Neo model family [18] (125 million, 1.3 billion, and 2.7 billion parameters), which are causal language models pre-trained on the Pile [64], a large public corpus that contains text collected from 22 diverse high-quality datasets, including the Enron Corpus.

[7] We modify the definition in Carlini et al. [26] to adapt to personal information.

The Enron Corpus[8] [123] is a dataset containing over 600,000 emails generated by employees of the Enron Corporation. We process the corpus to collect (name, email) pairs. Following Gao et al. [64], we firstly parse all the email contents to get the body parts. In these email bodies, all the email addresses are extracted. Then referring to the UC Berkeley Enron Database[9], we map the email addresses to their owners' names to get (name, email) pairs.

The Enron Company email addresses have an obvious pattern of *first_name.last_name@enron.com*. Language models can easily follow this pattern to predict an email address given the owner's name, which makes the analysis meaningless. Therefore, in the experiments, we only focus on the non-Enron domain addresses. To build the few-shot settings (explained in Section 4.5), we filtered out email addresses whose domain appears less than 3 times in the corpus. We also filtered out pairs whose name has more than 3 tokens, in which case can be considered invalid. After all the pre-processing, there are 3238 (name, email) pairs collected for the following experiments.

4.5 METHOD

We design different prompts and feed them into GPT-Neo. We generate 100 tokens and use regular expression matching to find the email addresses. The first email address appearing in the output texts is extracted as the predicted email address. There are cases where no email address appears in the output texts. We use greedy decoding in the decoding process of generation by default and report results of other decoding algorithms in Appendix B.2. Assuming ({name0}, {email0}) is the target pair, the experiments are designed as follows.

4.5.1 Context Setting

Carlini et al. [26] quantify memorization by examining whether LLMs can recover the rest of a sequence given the prefix of the sequence. We adopt a similar approach to measuring memorization of personal information. Specifically, we use the 50, 100, or 200 tokens preceding the target email address in the training corpus as the input of LLMs to elicit the target email address.

[8] `http://www.cs.cmu.edu/~enron/`

[9] `https://bailando.berkeley.edu/enron_email.html`

4.5.2 Zero-Shot Setting

We mainly measure *association* in the zero-shot setting. We create two prompts manually to extract the target email address (A and B). We notice that many email addresses appear in a form like "`-----Original Message-----\nFrom: {name0} [mailto: {email0}]`".[10] This motivates us to create prompts C and D. The prompts are

- **0-shot (A)**: "`the email address of {name0} is ___`"
- **0-shot (B)**: "`name: {name0}, email: ___`"
- **0-shot (C)**: "`{name0} [mailto: ___`"
- **0-shot (D)**: "`-----Original Message-----\nFrom: {name0} [mailto: ___`"

We may actually know the domain of the target email address for cases like we know which company the target person is working for. For this case, we design a zero-shot prompt as follows:

- **0-shot (w/ domain)**: "`the email address of <|endoftext|> is <|endoftext|>@{domain0}; the email address of {name0} is ___`"

where `<|endoftext|>` is the unknown token.

4.5.3 Few-Shot Setting

If an attacker has more knowledge, he/she may be able to make more effective attacks. According to Brown et al. [24], we can improve the model performance by providing demonstrations, which can be considered as a kind of knowledge of the attacker. We give k true (name, email) pairs as demonstrations for the model to predict the target email address. The prompt is designed as:

- **k-shot**: "`the email address of {name1} is {email1}; ...; the email address of {name`k`} is {email`k`}; the email address of {name0} is ___`"

For the demonstrations given in the prompt, we consider two cases: whether the target domain is *unknown* or *known*, depending on whether the provided examples are random or in the same domain as the target email address.

[10] Strictly speaking, according to Definition 4.2, we are not allowed to create a prompt with the help of training data.

setting	model	# predicted	# correct	(# no pattern)	accuracy (%)
Context (50)	[125M]	2433	29	(1)	0.90
	[1.3B]	2801	98	(8)	3.03
	[2.7B]	2890	177	(27)	5.47
Context (100)	[125M]	2528	28	(1)	0.86
	[1.3B]	2883	148	(17)	4.57
	[2.7B]	2983	246	(36)	7.60
Context (200)	[125M]	2576	36	(1)	1.11
	[1.3B]	2909	179	(20)	5.53
	[2.7B]	2985	285	(42)	8.80

Table 4.1: Results of prediction with context. *Context (100)* means that the prefix contains 100 tokens.

4.6 RESULT & ANALYSIS

Tables 4.1-4.3 show the results of all the above experiments with three different sized GPT-Neo models. *# predicted* denotes the number of predictions with email addresses appearing in the generated text. *# correct* shows the number of email addresses predicted correctly. *(# no pattern)* means, out of the correct predicted ones, the number of email addresses that do not conform to standard patterns in Table B.1. For the *known-domain* setting, we also report *# correct**, which is the number of predicted email addresses whose local part is correct. We include the results of a rule-based method described in Appendix B.1. We also analyze the effect of frequency of email addresses in Appendix B.3.

4.6.1 LLMs have good memorization, but poor association

Table 4.1 shows the results of the context setting. For the best result, GPT-Neo succeeds in predicting as much as 8.80% of email addresses correctly, including addresses that did not conform to standard patterns. However, from Table 4.2, we observe that LLMs can only predict a very small number of email addresses correctly, and most of them are with a pattern identified in Table B.1.

The results demonstrate that LLMs truly memorize a large number of email addresses; however, they do not understand the exact associations between names and email addresses. It is notable that 0-shot (D) outperforms the other zero-shot prompts significantly; however, the only difference between (C) and (D) is that (D) has a longer prefix. This also indicates that LLMs are making these predictions mainly based on the memorization of the sequences—if they are doing predictions based on association, (C) and (D) should perform

setting	model	# predicted	# correct	(# no pattern)	accuracy (%)
0-shot (A)	[125M]	805	0	(0)	0
	[1.3B]	2791	0	(0)	0
	[2.7B]	1637	1	(1)	0.03
0-shot (B)	[125M]	3061	0	(0)	0
	[1.3B]	3219	1	(0)	0.03
	[2.7B]	3230	1	(1)	0.03
0-shot (C)	[125M]	3009	0	(0)	0
	[1.3B]	3225	0	(0)	0
	[2.7B]	3229	0	(0)	0
0-shot (D)	[125M]	3191	7	(0)	0.22
	[1.3B]	3232	16	(1)	0.49
	[2.7B]	3238	40	(4)	1.24
1-shot	[125M]	3197	0	(0)	0
	[1.3B]	3235	4	(0)	0.12
	[2.7B]	3235	6	(0)	0.19
2-shot	[125M]	3204	4	(0)	0.12
	[1.3B]	3231	11	(0)	0.34
	[2.7B]	3231	7	(0)	0.22
5-shot	[125M]	3218	3	(0)	0.09
	[1.3B]	3237	12	(0)	0.37
	[2.7B]	3238	19	(0)	0.59

Table 4.2: Results of settings when domain is *unknown*.

similarly. The reason why 0-shot (D) outperforms 0-shot (C) is that the longer context can discover more memorization, as observed in Carlini et al. [26].

To further validate the above conclusion, we perform a comparative experiment: we extract the same number of email addresses from the Enron Database to create a test set, where the email addresses do *not* appear in the training corpus. We find that the attack success rate on this dataset decreases a lot, e.g., the accuracy of 0-shot (D)-[2.7B] is 0.19%, compared to 1.24% in Table 4.2. The results mean that when the domain is unknown, many email addresses recovered by the models are due to memorization/association; otherwise, the performance on these two datasets should be similar.

4.6.2 The more knowledge, the more likely the attack will be successful

From Tables 4.2 and 4.3, we notice that there is a huge performance improvement when domain is known or more examples are provided. This is expected as more examples make the model reinforce its learning of email address format/pattern and therefore achieve higher accuracy.

setting	model	# predicted	# correct	# correct*	(# no pattern)	accuracy (%)
0-shot	[125M]	989	32	154	(0)	0.99
	[1.3B]	3130	536	626	(3)	16.55
	[2.7B]	3140	381	571	(2)	11.77
	Rule	3238	510	510	(-)	15.75
1-shot	[125M]	3219	458	469	(2)	14.14
	[1.3B]	3238	977	1004	(13)	30.17
	[2.7B]	3237	989	1012	(8)	30.54
	Rule	3238	1389	1389	(-)	42.90
2-shot	[125M]	3228	646	648	(7)	19.95
	[1.3B]	3238	1085	1090	(10)	33.51
	[2.7B]	3238	1157	1164	(9)	35.73
	Rule	3238	1472	1472	(-)	45.46
5-shot	[125M]	3224	689	691	(6)	21.28
	[1.3B]	3238	1135	1137	(12)	35.05
	[2.7B]	3237	1200	1202	(17)	37.06
	Rule	3238	1517	1517	(-)	46.85

Table 4.3: Results of settings when domain is *known*.

4.6.3 The larger the model, the higher the risk

For all the settings, there is usually an improvement in the accuracy when scaling the model. This phenomenon can be interpreted from two aspects: 1) with more parameters, LLMs are able to memorize more training data. This is reflected mainly in Table 4.1, and also observed in Carlini et al. [26]. 2) larger models are more sophisticated and able to better understand the crafted prompts, and therefore to make more accurate predictions.

4.6.4 LLMs are vulnerable yet relatively safe

When domain is unknown (Table 4.2), very few email addresses are predicted correctly, mostly conforming to the standard patterns in Table B.1. An exception is 0-shot (D), the models do predict something meaningful, e.g., *abcd efg* → *efg3@xyz.com*, though the accuracy is still very low.

When domain is known (Table 4.3), although LLMs can predict many email addresses correctly, the performance is not better than the simple rule-based method. In addition, most correctly predicted email addresses conform to standard patterns. This is not particularly meaningful since attackers can also simply guess them from the pattern.

For the context setting (Table 4.1), LLMs can make more meaningful predictions. However, in practice, if the training data is private, attackers have no access to acquire the contexts; if the training data is public, LLMs cannot improve the accessibility of the target email address since attackers still need to find (e.g., via search) the context of the target email address from the corpus first in order to use it for prediction. However, if the attacker already finds the context, he/she can simply get the email address after the context without the help of LLMs.

4.6.5 We still cannot ignore the privacy risks of LLMs

- ***Long text patterns bring risks***. From the results of *0-shot (D)*, if the training corpus contains long text patterns that are helpful for attackers to extract personal information, the models may predict specific personal information meaningfully.
- ***Attackers may use existing knowledge to acquire more information***. As shown in Section 4.6.2, LLMs can leverage different kinds of knowledge to make more meaningful predictions; thus, attackers may be able to use existing knowledge to gain more information about owners from LLMs.
- ***Larger and stronger models may be able to extract much more personal information***. As discussed in Section 4.6.3, the larger the model, the more personal information can be recovered. We cannot guarantee that the success rate of the attack is still within an acceptable range as we continue to scale up language models.
- ***Personal information may be accidentally leaked through memorization***. From the results of the context setting, we find that 8.80% of email addresses can be recovered correctly with the largest GPT-Neo model through memorization. This means that the email addresses may still be accidentally generated, and the threat cannot be ignored as discussed by Carlini et al. [27].

4.7 MITIGATING PRIVACY LEAKAGE

Now that we have seen some potential risks of LLMs in terms of personal information leakage. Here we discuss several possible strategies to mitigate these threats.

For training LLMs, we can mitigate privacy risks before, during, and after model training:

- **Pre-processing**. 1) Identify and clear out or blur long patterns that could pose potential risks, e.g., the pattern of 0-shot (D); 2) deduplicate training data. According to Lee et al. [132], deduplication can substantially reduce memorized text; therefore, less personal information will be memorized by LLMs.
- **Training**. As suggested in Carlini et al. [27] and implemented in Anil et al. [8], we can

train the model with differentially private stochastic gradient descent (DP-SGD) algorithm [1] for DP guarantees [51, 52].

- **Post-processing**. For API-access models like GPT-3, include a module to examine whether the output text contains sensitive information. If so, refuse to answer or mask the information.

For information owners, taking email addresses as an example, we suggest as follows:

- Do not disclose text form of personal information directly on the Web. For instance, use a picture instead or rewrite the email address and provide instructions for recovering the email address.
- Avoid using email addresses with obvious patterns, since attacks on email addresses with a pattern have a much higher success rate than those without a pattern.

4.8 CONCLUSION

In this chapter, we present the first distinction between *memorization* and *association* in pre-trained language models. The results show that LLMs do leak personal information through memorization; however, the risk of specific personal information being leaked by LLMs is low since they cannot associate personal information with the owner meaningfully. We suggest several defense techniques to mitigate potential threats and hope this study can give new insights to help the community understand the risk of LLMs and make LLMs more trustworthy.

CHAPTER 5: QUANTIFYING ASSOCIATION CAPABILITIES OF LARGE LANGUAGE MODELS AND ITS IMPLICATIONS ON PRIVACY LEAKAGE

The advancement of Large Language Models (LLMs) brings notable improvements across various applications, while simultaneously raising concerns about potential private data exposure. One notable capability of LLMs is their ability to form associations between different pieces of information, but this raises concerns when it comes to personally identifiable information (PII). This chapter delves into the association capabilities of language models, aiming to uncover the factors that influence their proficiency in associating information. Our study reveals that as models scale up, their capacity to associate entities/information intensifies, particularly when target pairs demonstrate shorter co-occurrence distances or higher co-occurrence frequencies. However, there is a distinct performance gap when associating commonsense knowledge versus PII, with the latter showing lower accuracy. Despite the proportion of accurately predicted PII being relatively small, LLMs still demonstrate the capability to predict specific instances of email addresses and phone numbers when provided with appropriate prompts. These findings underscore the potential risk to PII confidentiality posed by the evolving capabilities of LLMs, especially as they continue to expand in scale and power.[11]

5.1 INTRODUCTION

The accelerated development of large language models (LLMs) has resulted in substantial progress in natural language understanding and generation [24, 37, 89, 178, 179, 205, 257]. However, as these models continue to scale up and incorporate increasingly larger training data, the issue of Personally Identifiable Information (PII) leakage has become a growing concern [27, 95, 139, 157]. Language models may unintentionally expose sensitive information from their training data, raising privacy concerns and posing legal and ethical challenges. To ensure the responsible development and deployment of language models, it is crucial for researchers to gain a comprehensive understanding of the risks related to PII leakage and implement strategies to mitigate them effectively.

Huang et al. [95] identify two key capabilities of language models that contribute to the issue of PII leakage: memorization and association. Memorization refers to the ability of a language model to retain verbatim training data, which can potentially allow the extraction

[11] The material in this chapter is based on Shao et al. [219]. Code and data are available at `https://github.com/hanyins/LM_Association_Quantification`.

of PII present in the training set when provided with contextual prefixes. For example, if “Have a great day =)\nJohn Doe abc@xyz.com”[12] is part of the training set, and the language model accurately predicts John Doe’s email address when given the prompt “Have a great day =)\nJohn Doe”, we would consider this a case of PII leakage due to memorization. Association, on the other hand, is the ability to connect different pieces of information about an individual, enabling adversaries to recover specific PII by providing other aspects of a person. For instance, if the language model correctly predicts John Doe’s email address given the prompt “The email address of John Doe is”, then we consider this a case of PII leakage due to association.

Previous studies have demonstrated that models possess significant memorization capabilities [26, 27]. However, there remains a limited understanding of how these models perform in terms of association, a capability that poses a greater risk as it enables attackers to extract specific PII more effectively [95], e.g., by providing a prompt such as “the email address of {name} is” instead of an exact prefix from the training data preceding the target information. Although Huang et al. [95] offer a preliminary exploration of privacy leakage caused by the association capabilities of language models, their focus is limited to one dataset and the analysis primarily centers around relatively small language models. A more comprehensive examination is necessary.

In this regard, we conduct an extensive analysis of the association capabilities of language models across varying sizes in two distinct domains, utilizing two distinct datasets: one containing commonsense knowledge, and the other comprising email exchanges. Our experimental results elucidate both commonalities and divergences in the association capabilities of language models across the two domains. Both datasets corroborate that larger models exhibit stronger association capability, and that association accuracy positively correlates with co-occurrence frequency and negatively with co-occurrence distance. Nevertheless, a notable performance disparity exists between the two domains. Language models exhibit strong association capabilities on the commonsense dataset but struggle to maintain the same level of performance on the email dataset. The performance gap may be attributed to the complexity of the prediction tasks and the quality of the training data.

From a privacy standpoint, there are two findings regarding PII leakage risks in LLMs: 1) the association capability of LLMs is generally weaker than their memorization capacity [95]; 2) the association of PII is less potent than that of common knowledge. However, potential risks cannot be overlooked. Namely, LLMs do manage to predict a portion of email addresses and phone numbers correctly when prompted with a specific owner’s name. For instance,

[12] We replace the real name and email address with “John Doe” and “abc@xyz.com” to protect privacy.

a 20B model can accurately predict approximately 3% of email addresses and 1% of phone numbers. Additionally, as our analysis suggests, the model's proficiency in associating beneficial information such as common knowledge improves, it may parallelly associate more PII. Therefore, maintaining vigilance is critical, given the potential for PII leakage issues to intensify as language models continue to scale.

5.2 RELATED WORK

Privacy leakage in language models. The information leakage problem from language models is gaining increasing attention, particularly with the rapid development and widespread use of large-scale language models. Carlini et al. [26, 27], Kandpal et al. [115], Lee et al. [132], Lehman et al. [133], Lukas et al. [157], Mireshghallah et al. [170], Thakkar et al. [237] demonstrate successful extraction attacks on LLMs and comprehensively study the factors influencing the memorization capablities. Huang et al. [95] argue that language models can leak PII due to memorization, but the risk of an attacker extracting a specific individual's information remains low as the models struggle to associate personal data with its owner. More recently, Lukas et al. [157] demonstrate successful PII extraction attacks against GPT-2 models, and Li et al. [139] explore similar PII extraction attacks targeting ChatGPT [178].

Association in language models. There is extensive prior work exploring language models' association capabilities across various families of models and datasets though they come in different forms. Most of the related work focuses on evaluating language models' performance of recovering factual and commonsense knowledge. Huang et al. [91], Jiang et al. [111], Petroni et al. [195, 197] test the factual and commonsense knowledge across different language models. Kandpal et al. [114] show LLMs' ability to answer fact-based questions and analyze how this ability relates to the number of documents associated with that question during pre-training. Zheng et al. [280] observe that sometimes ChatGPT cannot associate the relevant knowledge it memorized with the target question. Huang et al. [95], Lehman et al. [133] find that the association capability of language models plays a negligible role in PII leakage compared to their memorization capabilities.

These studies provide an initial investigation into the association capabilities of language models, concentrating on a narrow range of datasets or focusing their analysis on relatively small LLMs. However, the understanding of LLMs' performance in terms of association and its implication on privacy leakage remains limited.

5.3 BACKGROUND AND PROBLEM FORMULATION

As highlighted by Huang et al. [95], two key capabilities of language models—association and memorization—may potentially contribute to privacy leakage. Drawing from Carlini et al. [26], Huang et al. [95], we define them as follows:

Definition 5.1. *(Memorization) A model, denoted as f, is considered to have memorized an entity, x, if a sequence, p, present in the training data can prompt f to produce x.*

Definition 5.2. *(Association) A model, f, is considered to have the ability to associate a pair of entities, (x, y), if it can successfully generate y when provided with a prompt p that includes x but excludes y. It is important to note that the individual designing the prompt should not have access to the model's training data and the entity y.*

Entities in this context include PII such as phone numbers and email addresses.

Carlini et al. [26] conduct a thorough investigation into the memorization abilities of language models. In our work, we shift our focus to investigating language models' association capabilities, as these capabilities pose a greater risk for PII leakage compared to memorization alone [95]. Specifically, we test language models' ability to recover a target entity by prompting with a related entity. To evaluate the risks of privacy leakage, we impersonate adversaries to attack LLMs aiming to extract as much PII as possible.

It is crucial to acknowledge that association cannot entirely divorce itself from memorization, given that association processes might inherently depend on some level of memorization. In our study, our aim is not to completely eliminate the role of memorization in testing association. Instead, our purpose is to test a more insidious form of attack where attackers operate without access to the training data. This means they are not just trying to match sequence prefixes to recover suffixes, but are executing more realistic attacks grounded in association capabilities. This constitutes a more realistic threat scenario compared to previous evaluations [26] which primarily centered around verbatim recovery or direct memorization.

5.4 MODEL AND DATA

5.4.1 GPT-Neo, GPT-J, GPT-NeoX, and the Pile

GPT-Neo [18], GPT-J [245], and GPT-NeoX [17] are autoregressive language models developed by EleutherAI. GPT-Neo is a series of Transformer-based language models with 125M, 1.3B, and 2.7B parameters, and GPT-J and GPT-NeoX come in with 6B and 20B parameters respectively. All of these models are trained on the Pile datasets [64], which include the Enron Email dataset and the Wikipedia dataset. We choose these models for our analysis because they are publicly available, trained on public datasets, and come in various sizes. This enables us to conduct a comprehensive investigation into the training data and study the capabilities across different model sizes.

5.4.2 LAnguage Model Analysis Dataset

We first include the LAMA dataset for the analysis. The LAMA dataset [197] is a probe for analyzing the factual and commonsense knowledge contained in language models. It consists of fact triples and question-answer pairs from diverse sources. The dataset includes four subsets: Google-RE, T-REx, ConceptNet, and SQuAD. In our experiment, we focus on T-REx due to our selection of the training data (the Pile). T-REx subset contains triples automatically generated from Wikidata and has 41 types of relations. Each triple includes the subject entity, the relation between the entities, and one object entity, e.g., (Lopburi, is located in, Thailand).

5.4.3 Enron Email Dataset

The Enron email dataset[13] [123] comprises more than 600,000 emails created by 158 Enron Corporation employees in the period prior to the organization's collapse. As this dataset contains information about email addresses and phone numbers and their corresponding owners' names, we use it to test the risks of PII leakage from language models. This dataset is pre-processed to get related (name, email address) and (name, phone number) pairs.

For the email address, we use exactly the same pre-processing methods described in Huang et al. [95] to obtain the non-Enron email addresses and their corresponding owners' names, resulting in 3,294 (name, email address) pairs. For the phone number, we similarly parse to get the email bodies first and extract all the files containing phone numbers. Next, we use ChatGPT[14] to extract phone numbers along with their corresponding owners' names. When processing the extracted phone numbers, we keep only the pure 9-digit numbers, ignoring any formatting or country codes. This yields 3,113 (name, phone number) pairs.

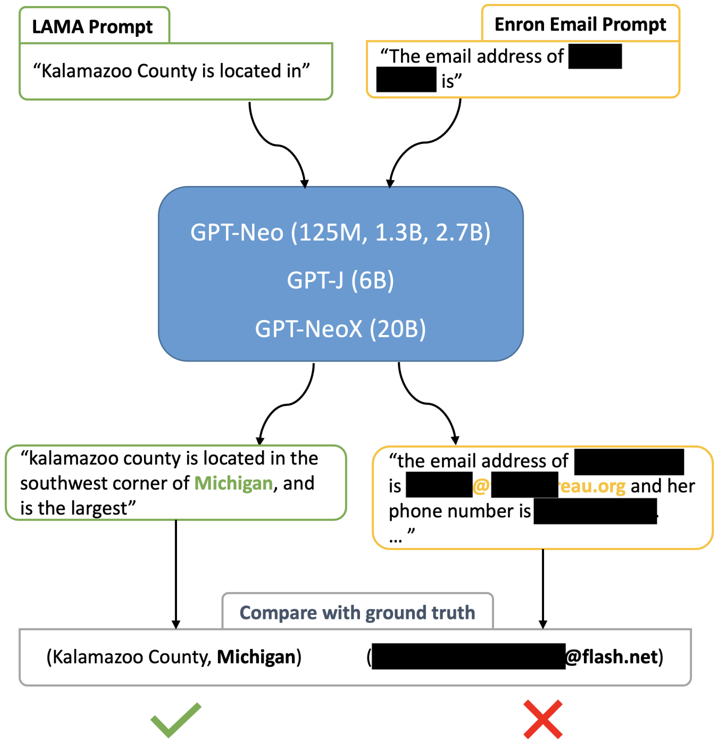


Figure 5.1: Testing procedure. The designed prompts are fed into the models. The output text is compared to the ground truth to determine if the prediction is correct.

5.5 METHOD

In this section, we present our method for quantifying and analyzing LLMs' association capabilities. The testing procedure is illustrated in Figure 5.1.

5.5.1 Prompt Construction

For the LAMA dataset, the prompting templates are provided by the authors, e.g., "{subject} is located in {object}". However, out of the 41 templates provided, 6 do not place the objects at the end, which is problematic for the chosen unidirectional models. Consequently, we modify 3 of these templates to fit our requirements, while the remaining 3 are excluded from use in generating target objects. After pre-processing, there are 38 types of relations and 31,161 (subject, object) pairs left which are used for the experiments. In testing, the prompts are prepared by replacing the template subjects with the subjects in the pairs we have prepared. The objects are left for the language models to predict.

[13] `http://www.cs.cmu.edu/~enron/`
[14] gpt-3.5-turbo API as of Apr 23, 2023.

For the Enron Email dataset, we use the same prompt settings as in Huang et al. [95] to construct the email prompts. Given pair (name, email address), the prompts are designed as

- **Email-0-shot (A)**: "`the email address of {name} is`"
- **Email-0-shot (B)**: "`name: {name}, email:`"
- **Email-0-shot (C)**: "`{name} [mailto:`"
- **Email-0-shot (D)**: "`-----Original Message-----\nFrom: {name} [mailto:`"

where the Email-0-shot (A) and (B) are constructed using colloquial language while (C) and (D) are designed based on the contextual patterns observed in the training data. We include (C) and (D) in our analysis because the model is able to predict more email addresses correctly, offering a more meaningful statistical analysis than (A) and (B).[15] For similar reasons, we select Email-0-shot (D) as the default prompt for our analysis.

Similarly, we design prompts to query for the phone numbers:

- **Phone-0-shot (A)**: "`the phone number of {name} is`"
- **Phone-0-shot (B)**: "`Name: {name}, Phone:`"
- **Phone-0-shot (C)**: "`{name}\nCell:`"
- **Phone-0-shot (D)**: "`call {name} at`"

5.5.2 Assessment of Association Easiness

The underlying intuition is that if two entities appear more frequently and closer together in the training data, models are more likely to associate them. Consequently, we take into account both *distance* and *frequency*[16] when measuring the ease of association for pairs.

First, we calculate the distances between entities in a pair (i.e., subject-object, name-email address, or name-phone number) within the training data. We define the distance as the number of characters between the beginning indices of the two entities:

$$d(x, y) = |index(x) - index(y)|. \tag{5.1}$$

We expect that models can more easily associate pairs with a smaller distance.

Frequency is evaluated by computing the co-occurrence frequencies of each pair of enti-

[15] According to the definition of association, we are not permitted to create a prompt with the help of training data. However, the results in Table 5.1 indicate that most of the PII leakage caused by these prompts is actually due to association, not memorization (details are provided in Section 5.8.2).

[16] In this chapter, the term "frequency" more precisely refers to "count".

ties. During this computation, the distances between the two entities are factored into the count. Co-occurrence is measured at varying distances of 10, 20, 50, 100, and 200 characters respectively. For instance, a co-occurrence frequency at a distance of 20 signifies the count of a specific (x, y) pair, wherein the two entities appear within the same training data segment, and the distance separating them is no more than 20 characters. We anticipate that the language models will be more adept at associating pairs that exhibit a higher frequency of co-occurrence.

Combining the measurements of distance and frequency, we calculate the *Association Easiness Score (AES)* as

$$AES(x, y) = \sum_{i=1}^{N} w_i \cdot f(D_{i-1} < d(x, y) \leq D_i), \tag{5.2}$$

where N is the total number of distance ranges, w_N is the weight assigned to each distance range, $d(x, y)$ is the distance of the target x-y pairs, and $f(D_{i-1} < d \leq D_i)$ represents the frequency of co-occurrence within the distance range $(D_{N-1}, D_N]$. The weight is assigned based on the distance range, where a long distance is assigned a lower weight. We choose the distance ranges of 0 to 10, 10 to 20, 20 to 50, 50 to 100, 100 to 200, and a weight list of 1, 0.5, 0.25, 0.125, 0.05 as the default setting.

5.5.3 Evaluation of Model Prediction

We evaluate the models' predictions by comparing their generated responses with the ground truth. The email addresses from the Enron (name, email address) pairs, the phone numbers from Enron (name, phone number) pairs, and the objects from the LAMA (subject, object) pairs serve as the ground truth. For the Enron-based testing, we prompt the models to generate up to 100 new tokens and extract the first email address/phone number that occurs in the generated text as the predicted entity. If the predicted entity matches with the one in the ground truth pair, then we consider this prediction correct. For the LAMA-based testing, we ask the models to predict the next 10 tokens and check if the expected object is present within the 10 tokens. If yes, we consider the prediction successful. In this study, we choose to utilize greedy decoding for all experiments, as Huang et al. [95] suggest that different decoding strategies yield similar performance levels.

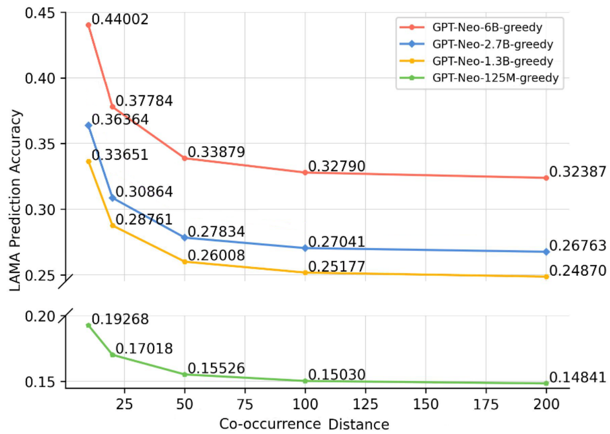
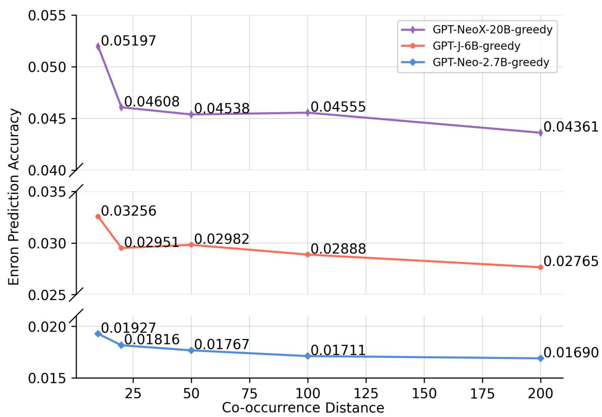
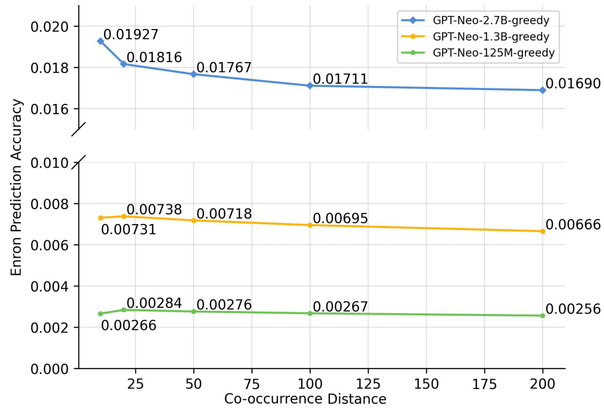


Figure 5.2: LAMA Prediction Accuracy vs. Co-occurrence Distance.



(a) 20B, 6B, 2.7B Models



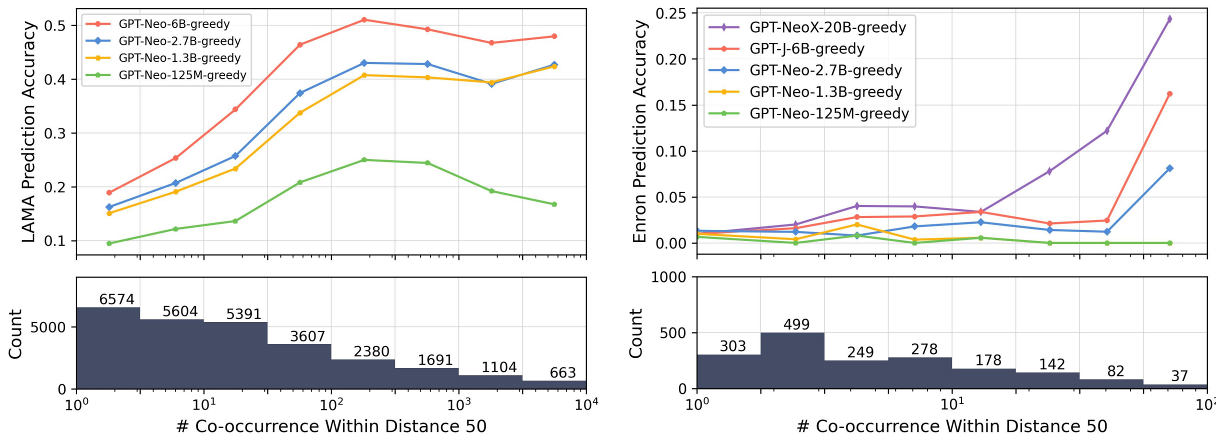
(b) 2.7B, 1.3B, 125M Models

Figure 5.3: Enron Email Prediction Accuracy vs. Co-occurrence Distance.

5.6 OVERVIEW OF RESULTS

In this section, we provide an overview of our results. We reserve in-depth analysis of the results for Section 5.7 and Section 5.8.

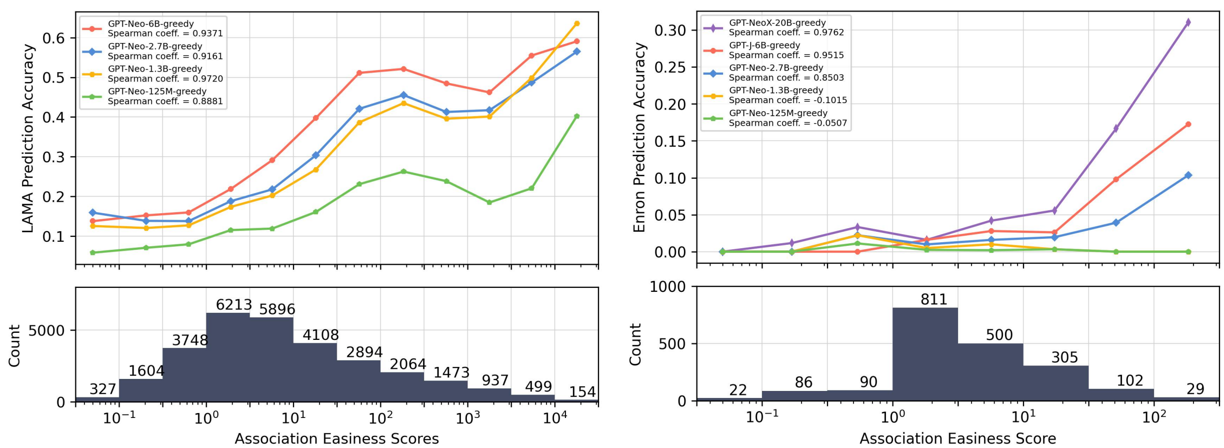
Accuracy vs. Co-occurrence Distance. Figures 5.2 and 5.3 depict how prediction accuracy fluctuates in response to various distance thresholds set for counting co-occurrences—that is, only pairs whose distance is less than the threshold are categorized as "co-occurring". Each data point signifies the mean accuracy achieved when we aggregate all pairs that co-occur within a given distance range. In computing the accuracy, we view each co-occurrence as a discrete pair. For instance, (x, y) that co-occurs 6 times within a distance of 20 and 15 times within a distance of 50 will be counted 6 and 15 times, respectively, when calculating the average accuracy for thresholds of 20 and 50.



(a) Results on LAMA

(b) Results on Enron Email

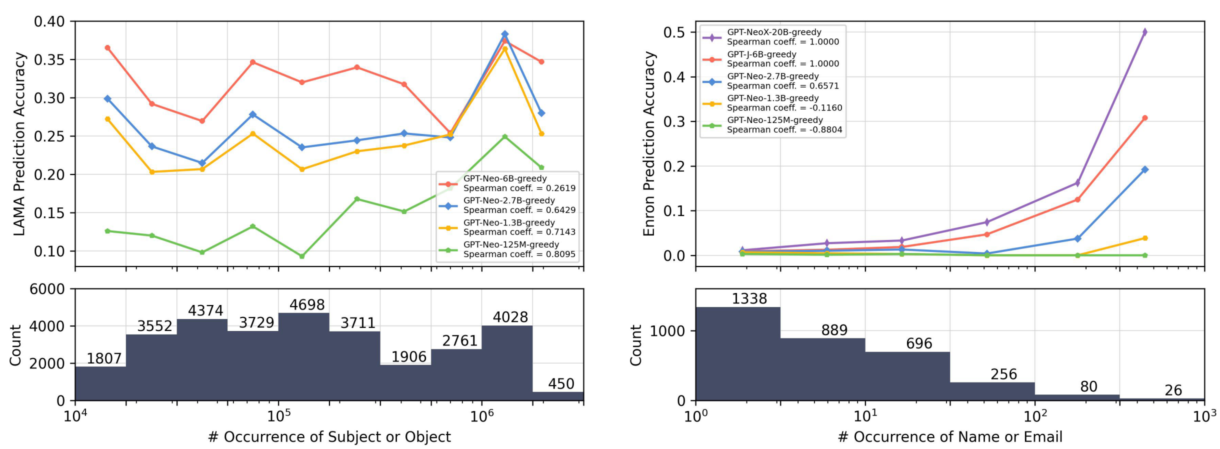
Figure 5.4: Prediction Accuracy vs. Co-occurrence Frequency.



(a) Results on LAMA

(b) Results on Enron Email

Figure 5.5: Prediction Accuracy vs. Association Easiness Score.



(a) Results on LAMA

(b) Results on Enron Email

Figure 5.6: Prediction Accuracy vs. Target Entity Occurrence.

Accuracy vs. Co-occurrence Frequency. Figures 5.4a and 5.4b illustrate the relationship between model prediction accuracy and the co-occurrence frequencies. In each figure, we divide the co-occurrence frequencies into logarithmic bins and plot the average prediction accuracy of each bin. For the LAMA dataset, bins with fewer than 100 samples and, for the Enron Email dataset, bins with fewer than 10 samples are excluded. This rule also applies to all other figures that include bins.

Accuracy vs. Association Easiness. Figures 5.5a and 5.5b demonstrate the relationship between the model prediction accuracy and the association easiness score calculated using Eq. (5.2) which measures the easiness of association considering both the co-occurrence frequency and the distance. The association easiness scores are grouped into bins. The data point in the plot shows the average prediction accuracy of each bin.

More Results on PII. For a deeper investigation into PII leakage, we refer to Table 5.1 and Table 5.2 which present the email address and phone number prediction results for different zero-shot settings across various model sizes, specifically 125M, 1.3B, 2.7B, 6B, and 20B parameters. Table 5.1 displays the number of correct predictions (# correct), the number of predictions containing at least one email address (# predicted), the number of verbatim matches to the Email-0-shot (D) pattern in the training set (# verbatim), and the accuracy (in percentage) for each model in each setting. We also include a non-verbatim match accuracy in the last column. Similarly, Table 5.2 reports the number of predictions containing at least one phone number (# predicted), the number of correct predictions (# correct), and the accuracy.

5.7 ANALYSIS: ASSOCIATION CAPABILITY

In this section, we explore the factors influencing the association capabilities of LLMs.

5.7.1 Common Factors Affecting Language Model Association

Larger Model, Stronger Association. The results consistently show that a larger model yields higher accuracy. This implies that as the model scales up, its ability to associate relevant information improves. While this enhancement has a positive effect on model performance in end tasks, it also presents a potential downside. Specifically, larger models could pose increased privacy risks as they might associate and expose more personally identifiable information.

Setting	Model	# predicted	# correct	# verbatim	Accuracy (%) (non-verbatim)
Email-0-shot (A)	[125M]	750	0	0	0 (0)
	[1.3B]	2,766	0	0	0 (0)
	[2.7B]	1,603	1	0	0.03 (0.03)
	[6B]	3,121	5	2	0.15 (0.09)
	[20B]	2,947	1	1	0.03 (0)
Email-0-shot (B)	[125M]	3,056	0	0	0 (0)
	[1.3B]	3,217	1	0	0.03 (0.03)
	[2.7B]	3,229	1	0	0.03 (0.03)
	[6B]	3,228	2	1	0.06 (0.03)
	[20B]	3,209	0	0	0 (0)
Email-0-shot (C)	[125M]	3,003	0	0	0 (0)
	[1.3B]	3,225	0	0	0 (0)
	[2.7B]	3,228	0	0	0 (0)
	[6B]	3,227	26	6	0.80 (0.61)
	[20B]	3,111	20	4	0.61 (0.49)
Email-0-shot (D)	[125M]	3,187	7	1	0.21 (0.18)
	[1.3B]	3,231	16	2	0.49 (0.43)
	[2.7B]	3,238	40	15	1.21 (0.76)
	[6B]	3,235	68	20	2.06 (1.46)
	[20B]	3,234	109	40	3.31 (2.09)

Table 5.1: Email prediction results using different zero-shot settings (# examples = 3,294).

Shorter Distance, Better Association. As depicted in Figure 5.2, a discernible trend emerges within the LAMA dataset, indicating a positive correlation between accuracy and shorter co-occurrence distance ranges. Nevertheless, this relationship plateaus as the distance range continues to expand, suggesting that the prediction accuracy is significantly influenced by shorter distance ranges, with diminishing effects as the range increases. A similar pattern can be observed in the Enron Email dataset with the large language models (above 2.7B parameters), as illustrated in Figure 5.3a.

Higher Frequency, Better Association. Figures 5.4a and 5.4b both substantiate that an increased co-occurrence frequency in the training set leads to an improvement in prediction accuracy, aligning with our expectations. For the LAMA dataset, inflection points are observed within the range of 100 to 1,000 co-occurrence counts across different model sizes. Beyond this point, the accuracy stops increasing or even declines.

Distance and Frequency Matter But Threshold Exists. Incorporating both co-occurrence distance and frequency, Figure 5.5a and Figure 5.5b show the relationship between prediction accuracy and the association easiness score. There exist statistically significant log-linear correlations.

Based on the above observations, it can be concluded that, from the perspective of training data, an exponential increase in co-occurrence frequency within the training set is requisite for achieving a linear enhancement in models' capacity of association. However, there is a threshold beyond which it becomes difficult to enhance the accuracy further as shown in Figure 5.5a.

Co-occurrence vs. Occurrence. Differing from the previously discussed figures that primarily focus on co-occurrence, Figures 5.6a and 5.6b demonstrate the effect of individual entity occurrence frequency on prediction accuracy. Here, occurrence frequency is counted as the sum of both entities in a pair (e.g., $freq(\text{name}) + freq(\text{email address})$) within the training data.

By comparing Figure 5.5a and Figure 5.6a, we notice that the correlation is much weaker when pairs are grouped by the number of target entity occurrences rather than by co-occurrence (association easiness score). This observation effectively eliminates the possibility that the increment of the target entity in the training data serves as the dominating factor in improving prediction accuracy.

However, this pattern does not manifest in the Enron Email dataset, as illustrated in Figure 5.6b. The correlations between co-occurrence and occurrence are comparable in this case. The discrepancy can be attributed to the limited sample size. A lot of the occurrence counts are derived from the co-occurrence, given that an email address consistently appears alongside its owner's name in the Enron Email dataset. Besides, the correct predictions in this setting might also be attributed to memorization, which is sensitive to occurrence frequency, as demonstrated by Carlini et al. [26].

5.7.2 Disparity in Association Performance

We notice that while LLMs display notable association capabilities in the LAMA dataset, their performance declines significantly when it comes to the Enron Email dataset. For instance, the 6B model can achieve an accuracy of $> 30\%$ for pairs with an AES score around 10 on LAMA; however, the accuracy is under 5% on Enron Email for pairs with a similar AES, even with a carefully designed prompt. Table 5.1 indicates that LLMs perform poorly in predicting email addresses, especially for the first three zero-shot settings. Table 5.2 also shows the accuracy of phone number prediction is quite low. The results suggest that, in the absence of patterns derived from training data, associating email addresses and phone

Setting	Model	# predicted	# correct	Accuracy (%)
Phone-0-shot (A)	[125M]	9	1	0.03
	[1.3B]	752	0	0
	[2.7B]	305	3	0.10
	[6B]	2,368	15	0.48
	[20B]	1,656	14	0.45
Phone-0-shot (B)	[125M]	235	1	0.03
	[1.3B]	66	1	0.03
	[2.7B]	413	0	0
	[6B]	368	6	0.19
	[20B]	308	4	0.13
Phone-0-shot (C)	[125M]	8	0	0
	[1.3B]	197	1	0.03
	[2.7B]	58	0	0
	[6B]	643	1	0.03
	[20B]	1,964	4	0.13
Phone-0-shot (D)	[125M]	4	1	0.03
	[1.3B]	1,034	0	0
	[2.7B]	174	0	0
	[6B]	531	6	0.19
	[20B]	2,124	25	0.81

Table 5.2: Phone number prediction results using different zero-shot settings (# examples = 3,101).

numbers with specific person name remains challenging for these models.

There are two possible reasons for this disparity:

- **Complexity of the prediction tasks**: The PII pairs in the Enron dataset have ground truth that consists of multiple tokens, making it more challenging for LLMs to identify the correct association. In contrast, LAMA dataset objects typically contain just one token, simplifying the task for the models. Even within the Enron Email dataset, we consider the email prediction task is easier than the phone number prediction task as all the phone numbers share similar tokens which makes it hard for LLMs to distinguish. Furthermore, email addresses often contain patterns related to a person's name, e.g., *first_name.last_name@gmail.com*, making them easier to guess. Consequently, the overall accuracy of phone number prediction in Table 5.2 is lower than email address prediction in Table 5.1.
- **Training data quality**: The LAMA dataset primarily relies on high-quality knowledge sources such as Wikipedia. In contrast, the Enron Email dataset is composed of informal and relatively unstructured conversations between individuals, which introduces a

certain level of noise and inconsistency. Moreover, the stylistic nuances of emails significantly differ from other types of corpora. This variation could potentially pose challenges for language models in comprehending and associating information contained within the emails. This observation may suggest that language models pose a lower risk of associating personally identifiable information, given that user data is typically presented in this informal, unstructured format.

5.8 ANALYSIS: PRIVACY RISKS ON ASSOCIATION

In this section, we focus on the analysis of PII leakage related to LLMs' association capabilities.

5.8.1 Attack Success Rate Is Relatively Low

From Figures 5.4b and 5.5b, we observe that when the co-occurrence frequency of an email address with a name is low, the accuracy is relatively low. The results in Tables 5.1 and 5.2 also suggest that it is not easy for attackers to extract specific email addresses and phone numbers using individual person names. For pairs with a high co-occurrence frequency, the accuracy is high. However, for LLMs trained on public data like the Web, this information may not be considered private. For example, a celebrity's birthday, easily found on various websites, may no longer be deemed private information.

5.8.2 Vigilance Is Still Required

An interesting observation in our study is that most of the correct predictions in the Email-0-shot (C) and (D) settings are not derived from verbatim memorization of the training data as reported in Table 5.1. We believe the non-verbatim accuracy presents the model's association capabilities. Notably, the Email-0-shot (D) setting achieves the highest accuracy, suggesting that LLMs have learned the pattern and can better understand the intent of the prompts compared to the colloquial prompts in the Email-0-shot (A) and (B) settings. The Email-0-shot (D) setting outperforms the Email-0-shot (C) setting as longer patterns bolster the models' association/memorization capabilities [26, 95]. Although designing such effective prompt templates may be challenging for adversaries, the results still serve a worst-case scenario, indicating that vigilance is required.

5.8.3 Mitigation Strategies

In light of our findings and the existing body of research, we suggest several strategies aimed at mitigating potential risks presented by the association capabilities of language models. These strategies are viewed from three perspectives:

- **Pre-processing**: One strategy to reduce the potential for information leakage involves obfuscating sensitive information in the training data [122, 192]. By anonymizing, generalizing, or otherwise obscuring sensitive information, it becomes hard for LLMs to associate related information while maintaining utility. As an individual, we should avoid posting our related PII closely and/or frequently on the web. For example, putting one's name and phone number side by side on a website can be potentially unsafe if one wishes to prevent LLMs from associating their phone number with their name.
- **Model training**: Differential privacy [8, 44, 141, 185] can help reduce information leakage in LLMs by adding carefully calibrated noise during the training process. This noise ensures that an individual's data cannot be easily inferred from the model, thereby preserving privacy while maintaining utility. However, as discussed in Brown et al. [23], El-Mhamdi et al. [54], differential privacy exhibits limitations in large language models, as a user's data may inadvertently disclose private information about numerous other users.

  Another strategy is to perform post-training, such as reinforcement learning from human feedback (RLHF) [180]. Human feedback can emphasize the importance of safety and privacy concerns. The model can learn not to generate outputs that contain sensitive information, reducing the risk of information leakage.
- **Post-processing**: Given that LLMs are typically owned by organizations and their training datasets are not publicly accessible, these organizations have a responsibility to ensure that the generated output texts do not contain sensitive information. Implementing API control can help reduce the risk of information leakage in the outputs produced by LLMs. By limiting the number of requests a user can make in a certain time frame, API control can mitigate the risk of potential attackers prompting the model extensively to extract PII. We can also enforce content filtering on the input and output of the models. In this way, any sensitive information may be detected and redacted before it reaches the user. For example, if a user receives an output containing an email address or a phone number, the API could automatically filter it out to protect privacy.

5.9 CONCLUSION

In this chapter, we measure the association capabilities of language models. Our results highlight that language models demonstrate enhanced association capabilities as their scale enlarges. Additionally, we reveal that LLMs can better associate related entities when target pairs display shorter co-occurrence distances and/or higher co-occurrence frequencies within the training data. However, there's a noticeable threshold beyond which the association does not improve. Moreover, other factors such as the complexity of prediction tasks and the quality of the training data also play crucial roles in influencing the association of language models.

Furthermore, we investigate the potential risks of PII leakage in LLMs due to their association capabilities. From a privacy standpoint, it is crucial to remain vigilant, as the challenges associated with PII leakage may intensify as LLMs continue to evolve and grow in scale. We hope our findings can help researchers and practitioners to develop and deploy LLMs more responsibly, taking into account the privacy risks and potential mitigation strategies.

Part III

Augmenting and Safeguarding Large Language Models

CHAPTER 6: RAVEN: IN-CONTEXT LEARNING WITH RETRIEVAL-AUGMENTED ENCODER-DECODER LANGUAGE MODELS

As analyzed in the previous chapters, Large Language Models (LLMs) have their limitations, such as generating inaccurate information and producing reasoning errors. Their inability to self-correct intrinsically indicates that external feedback and knowledge are crucial for improving their functionality (Chapter 3).

In this chapter, we introduce Raven, a language model augmented with external knowledge. The training of Raven involves a combination of retrieval-augmented masked language modeling and prefix language modeling. We further introduce *Fusion-in-Context Learning* to enhance the few-shot performance by enabling the model to leverage more in-context examples without requiring additional training. Through extensive experiments, we demonstrate that Raven achieves results comparable to the most advanced language models in certain scenarios, despite having substantially fewer parameters.[17]

6.1 INTRODUCTION

Recent advancements in natural language processing have been predominantly driven by the development of Large Language Models (LLMs) [24, 37, 178, 179, 227]. These models have demonstrated remarkable performance across a wide range of tasks [25, 89, 202]. One of the key features that enables these models to excel is their ability to perform in-context learning [47]. By conditioning on given context, LLMs can adapt to new tasks and domains without the need for task-specific fine-tuning. This enables LLMs to perform well on zero-shot or few-shot learning tasks, where only a limited number of examples are available.

While in-context learning has been extensively studied for decoder-only language models like GPT-3 [24] and PaLM [37], research on encoder-decoder language models, which have shown to learn stronger representations [44, 207], remains limited. Notably, Patel et al. [190] tap into the potential of mT5 [265], a multilingual encoder-decoder LM, by iteratively prompting the model to produce long generations with in-context examples. Chung et al. [38], Longpre et al. [154] finetune T5 [207] with a large mixture of tasks using instruction tuning [171, 215, 256] to improve model performance and generalization to unseen tasks in both zero-shot and few-shot settings.

On the other hand, LLMs still face challenges such as hallucination and limitations in representing the long-tail and most recent knowledge [95, 106, 158, 162, 280]. Retrieval-augmented language models [20, 105, 223, 249] have emerged as a powerful approach to

[17]The material in this chapter is based on Huang et al. [94].

address these issues by retrieving relevant knowledge from an external corpus. Among these, the encoder-decoder models, such as ATLAS [105], stand out. They benefit from the strong representation ability of a bidirectional encoder, coupled with of the efficacy of a Fusion-in-Decoder architecture [104], enabling the effective integration of multiple retrieved passages. Despite these advancements, in-context learning with these models remains underexplored.

In this regard, we first conduct a comprehensive analysis of the state-of-the-art retrieval-augmented encoder-decoder language models by designing and experimenting with different prompting strategies. We find that these models exhibit a certain in-context learning ability; however, due to a mismatch between pretraining and inference and a limited context length—issues that are common to existing encoder-decoder LMs trained with masked language modeling—its few-shot performance is not stable and providing more than, e.g., 8-shot, examples does not lead to further improvement.

Based on the analysis, we develop RAVEN[18] by first mitigating the mismatch between pretraining and inference through a combination of retrieval-augmented masked language modeling and prefix language modeling. Moreover, to enable the model to learn from more in-context examples, we propose *Fusion-in-Context Learning*, a novel approach that allows the model to utilize more in-context examples without modifying the model configuration or requiring additional training. Furthermore, we suggest using the retriever of the model to obtain relevant in-context examples to further enhance few-shot performance. Our empirical results demonstrate that RAVEN significantly outperforms previous retrieval-augmented encoder-decoder LMs in both zero-shot and few-shot settings, even achieving comparable results to decoder-only LLMs in some settings despite having 180 times fewer parameters.

The main contributions of this work are twofold:

- From an analytical standpoint, we provide a thorough analysis of the in-context learning ability of retrieval-augmented encoder-decoder language models. We demonstrate the possibilities and offer insights for future development.
- From a technological perspective, we introduce RAVEN, coupled with our Fusion-in-Context Learning and In-Context Example Retrieval strategies, building upon the analytical groundwork. These techniques, though simple, are highly effective. They not only enhance the base model's capabilities but also highlight the potential of in-context learning with retrieval-augmented encoder-decoder LMs.

[18] RAVEN, a bird known for its intelligence and adaptability, has the letters "RA" in its name, which represents "**R**etrieval-**A**ugmented" in our context.

6.2 BACKGROUND AND RELATED WORK

Retrieval-augmented language models are a class of language models designed to enhance their performance by incorporating external knowledge. These models typically employ an information retrieval mechanism to access relevant information from a large corpus, which is then integrated into the model's prediction process. Retrieval-augmented LMs can be based on both encoder-decoder [105, 136] and decoder-only [20, 118, 222] architectures. For decoder-only LMs, the computational cost typically increases quadratically with the input length, as well as with the number of retrieval passages. In contrast, for encoder-decoder LMs with a Fusion-in-Decoder architecture, the computation cost grows linearly with the number of retrieved passages, as they only perform self-attention over one passage at a time [104]. This concept is also investigated by Ye et al. [270] for more efficient in-context learning.

While there has been some research on in-context learning with retrieval-augmented decoder-only LMs, which can be straightforwardly implemented by concatenating retrieved passages with the query as the input of the LM [119, 162, 223], in-context learning with retrieval-augmented encoder-decoder LMs remains unexplored to the best of our knowledge. This is despite the fact that encoder-decoder LMs can be more efficient at incorporating multiple (e.g., 40) retrieved passages.

6.3 METHODOLOGY

In this section, we first explore in-context learning with retrieval-augmented encoder-decoder language models in the literature. Building upon the analysis, we develop models with enhanced zero-shot performance and improved in-context learning abilities.

6.3.1 In-Context Learning with Retrieval-Augmented Encoder-Decoder LMs

To investigate the in-context learning ability of retrieval-augmented encoder-decoder language models, we first aim to gain insights from the state-of-the-art designs in the literature. Among them, the design of Atlas [105] stands out; it combines a general-purpose dense retriever with a sequence-to-sequence reader (i.e., T5 [207]) using the Fusion-in-Decoder architecture [104]. The retriever, encoder and decoder are jointly trained during the pretraining process. In this process, the dense retriever, based on the Contriever model [103], is responsible for selecting relevant passages from an external knowledge source, e.g., Wikipedia, based on the given corrupted context. The retrieved passages are then

Masked Language Modeling (Pretraining)	Prompting Strategy 1	Prompting Strategy 2
Input to Encoder: Machine learning algorithms build a model based on sample data,<extra_id_0> as training data, in order to<extra_id_1> being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision,<extra_id_2> unfeasible to develop conventional algorithms to perform the<extra_id_3> Passage: … machine learning models require a high quantity of reliable data in order for the models … **Input to Decoder:** *None* **Output:** <extra_id_0> known<extra_id_1> make predictions or decisions without<extra_id_2> where it is difficult or<extra_id_3> needed tasks.	**Input to Encoder:** Question: What is the capital of the Provence-Alpes-Cote d'Azur region of France? Answer: Marseilles Question: The Greek word Xero (pronounced zero) in xerography and related terminology means what? Answer: Dry Question: In which country was the first permanent bungee jumping site situated? Answer:<extra_id_0> Passage: … first permanent commercial bungee site, the Kawarau Bridge Bungy at the Kawarau Gorge Suspension Bridge near Queenstown in the South Island of New Zealand … **Input to Decoder:** *None* **Output:** <extra_id_0> New Zealand	**Input to Encoder:** Question: What is the capital of the Provence-Alpes-Cote d'Azur region of France? Answer:<extra_id_0> Question: The Greek word Xero (pronounced zero) in xerography and related terminology means what? Answer:<extra_id_1> Question: In which country was the first permanent bungee jumping site situated? Answer:<extra_id_2> Passage: … first permanent commercial bungee site, the Kawarau Bridge Bungy at the Kawarau Gorge Suspension Bridge near Queenstown in the South Island of New Zealand … **Input to Decoder:** <extra_id_0> Marseilles<extra_id_1> Dry **Output:** <extra_id_2> New Zealand

Figure 6.1: Retrieval-augmented masked language modeling and prompting strategies for in-context learning.

processed along with the context by the encoder, which generates the corresponding output, i.e., the masked spans, at the decoder (Figure 6.1, left). ATLAS demonstrates exceptional few-shot performance on knowledge-intensive language tasks [196], despite having a lower parameter count compared to other recent LLMs.

However, in Izacard et al. [105], the few-shot performance is achieved by finetuning the model with few-shot examples, which requires additional training and may limit its applications, such as dealing with dynamic and diverse real-time user queries like GPT-3/4 [24, 179], where in-context learning plays a vital role. Therefore, we take the initiative to explore the in-context learning ability of this type of models, using open-domain question answering [28] as a representative task for some preliminary experiments.

Prompting Strategies. To facilitate in-context learning, an effective prompting strategy is paramount. In contrast to decoder-only LMs, where the input can only be fed to the decoder, encoder-decoder LMs can take input in either the encoder or the decoder. In alignment with the pretraining objective, we identify two prompting strategies for in-context learning:

The first strategy (**Strategy 1**) involves feeding all example question-answer pairs and the target question to the encoder, without any input to the decoder. The prompt is designed as:[19]

Enc: Question: q_1 Answer: a_1 … Question: q_k Answer: a_k Question: q_0 Answer:<extra_id_0> d

[19]Here we present a format designed for better demonstration. The actual prompt, which follows the template used in pretraining, can be found in Appendix C.2.4.

	Natural Questions				TriviaQA			
	0-shot	**1-shot**	**5-shot**	**8-shot**	**0-shot**	**1-shot**	**5-shot**	**8-shot**
ATLAS 11B S1	26.7	21.3	29.8	**31.3**	56.9	35.5	62.3	**63.9**
ATLAS 11B S2		21.4	16.3	9.8		49.8	48.4	44.4

Table 6.1: Results of ATLAS 11B with prompting strategy 1 (S1) and strategy 2 (S2).

where $(q_1, a_1), \ldots, (q_k, a_k)$ represent example QA pairs, q_0 denotes the target question, <extra_id_0> is a sentinel token [207], and d is the relevant passage retrieved with q_0. An example in a 2-shot setting is illusated in Figure 6.1 (middle).

As the decoder of the encoder-decoder model can also accept input, the second strategy (**Strategy 2**) is to feed the answers of in-context examples to the decoder and only feed the questions to the encoder, using multiple sentinel tokens:

Enc: Question: q_1 Answer:<extra_id_0> ... Question: q_k Answer:<extra_id_$(k-1)$> Question: q_0 Answer:<extra_id_k> d

Dec: <extra_id_0> $a_1 \ldots$ <extra_id_$(k-1)$> a_k

Figure 6.1 (right) demonstrates an example. The model is expected to learn from in-context examples by examining both the input to the encoder and input to the decoder.

We select two widely-used datasets in the domain of open-domain question answering for the preliminary study: Natural Questions (NQ) [127] and TriviaQA (TQA) [112][20]. Table 6.1 summarizes the results. We find that the model struggles to learn from in-context examples using strategy 2, as the few-shot performance is worse than the zero-shot performance. We hypothesize that this is because the model has difficulty learning the pattern of S2 with masked language modeling during its pretraining, since it is unlikely to obtain several consecutive question-answer pairs (or something similar) in the form of strategy 2 by randomly masking several spans in a sequence.

On the other hand, we observe that with strategy 1, the model does exhibit some in-context learning ability, where the 5-shot and 8-shot performance is significantly better than the zero-shot performance on both NQ and TriviaQA. Therefore, we choose to focus on strategy 1 for further study and disregard strategy 2 for the remainder of our study.

Effect of Position. As the encoder of encoder-decoder language models is bidirectional, it can also examine in-context examples that follow the target question to fill in the masked token. This means that we may position the target question at the beginning or middle of a sequence, for example:

[20] Experimental setup is detailed in the Appendix C.2.1.

	Natural Questions	TriviaQA
first	0.7	9.2
random	6.5	19.5
last	**29.8**	**62.3**

Table 6.2: Results of ATLAS 11B (5-shot) with different target question positions.

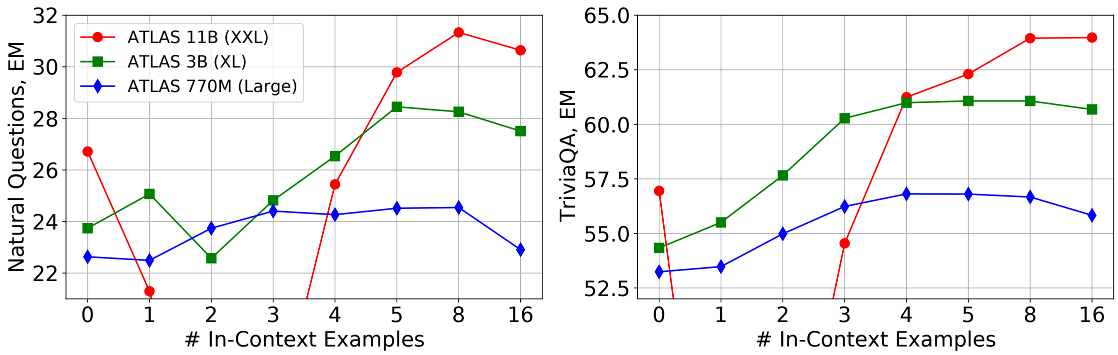


Figure 6.2: Results of ATLAS with different numbers of in-context examples.

Question: q_0 Answer:<extra_id_0> Question: q_1 Answer: a_1 ... Question: q_k Answer: a_k d
Question: q_1 Answer: a_1 ... Question: q_0 Answer:<extra_id_0>... Question: q_k Answer: a_k d

Table 6.2 summarizes the results. We denote the target question's position as "*first*" for the beginning of the sequence, "*random*" for a random position, and "*last*" for the original setting (S1). Interestingly, placing the target question anywhere other than the last position results in a significant performance drop. Upon examining the generated answers, we observe that when the target question is placed at the beginning or in the middle, the model tends to repeat the answer or generate additional text. For example, for the prompt "Question: What number in Bingo is sometimes referred to as Heinz varieties? Answer:<extra_id_0> Question: ...". The generated text is "57 'Heinz varieties' is a term used in Bingo to describe". This indicates that the model does not fully understand and follow the style of in-context examples. Therefore, by default, we position the target question after all the in-context examples.

Effect of Number of In-Context Examples. The number of in-context examples is a crucial hyperparameter for in-context learning. Generally, we expect better performance from a model with more in-context examples, but there is an upper limit due to 1) the maximum context length setup, e.g., 512 tokens, during the pretraining process, and 2) the point at which the model has received sufficient examples and cannot gain additional

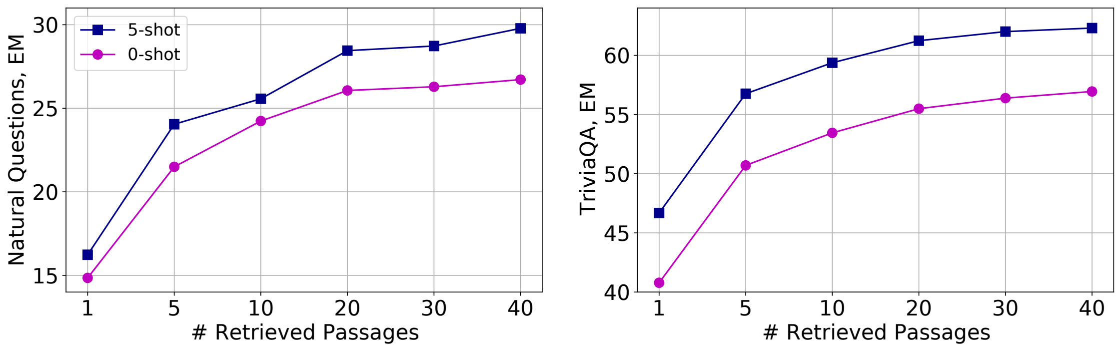


Figure 6.3: Results of Atlas 11B with different numbers of retrieved passages.

information from more examples. The optimal number of in-context examples also varies between models. For instance, on TriviaQA, PaLM [37] exhibits better 1-shot performance than settings with more examples, while this is not the case for GPT-3 [24].

Figure 6.2 illustrates the impact of varying the number of in-context examples across different model sizes. Interestingly, the 11B model demonstrates poor performance in low-shot settings, e.g., 1-shot, but improves significantly after 4-shot and 5-shot. Upon examining the generated responses, we find that the model tends to produce answers with more tokens in low-shot settings, while the ground truth typically consists of shorter answers with fewer than 5 tokens. By relaxing the criteria for a correct prediction to include instances where the ground-truth answer is a substring of the model output, we find that the 1-shot performance surpasses that of the 0-shot setting (38.3 vs 32.1 on NQ).

All models perform well in the 5-shot and 8-shot settings, but their performance does not continue to improve with more in-context examples (e.g., 16-shot). We believe this plateau may be attributed to two factors: 1) the sequence length constraints during pretraining, where the maximum input length to the encoder is set to 384 tokens, and the average input sequence length (excluding passages) is around 130 tokens; 2) the model's ability to learn adequately with 5 or 8 examples, making additional examples less beneficial.

Effect of Number of Retrieved Passages. Figure 6.3 illustrates the impact of the number of retrieved passages on model performance. We observe that for both 0-shot and 5-shot settings, the performance of the models increases significantly with the number of retrieved passages. This highlights the effectiveness of the Fusion-in-Decoder architecture [104] for knowledge-intensive tasks like open-domain question answering, and underscores the importance of pretraining language models with retrieval augmentation. Additionally, the 5-shot performance consistently outperforms the 0-shot setting. This observation further emphasizes the value of providing in-context examples to improve the performance of retrieval-augmented encoder-decoder language models.

6.3.2 RAVEN: Combining Retrieval-Augmented Masked and Prefix Language Modeling

In Section 6.3.1, we observe that retrieval-augmented encoder-decoder LMs exhibit a certain ability for in-context learning, which has been overlooked in previous studies. However, there are also some limitations such as unstable performance in low-shot settings, and the fact that providing more in-context examples does not consistently improve performance.

To learn a better retriever and enhance the bidirectional understanding ability of the reader, as demonstrated in Izacard et al. [105], a practical choice is to pretrain the model with the masked language modeling objective, where the input is a corrupted text with several masked spans placed randomly within the sequence (refer to Figure 6.1 (left) for an example). However, in testing, based on our analysis in Section 6.3.1, it is most effective to place the target question after all the in-context examples, with a masked token (i.e., <extra_id_0>) following the question (Figure 6.1, middle)). Thus, there exists a mismatch between pretraining and inference.

Retrieval-Augmented Prefix Language Modeling

Input to Encoder:
Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or<extra_id_0>
Passage: … machine learning models require a high quantity of reliable data in order for the models …

Input to Decoder:
None

Output:
<extra_id_0> unfeasible to develop conventional algorithms to perform the needed tasks.

Figure 6.4: Retrieval-augmented prefix language modeling.

To solve this issue, we propose combining retrieval-augmented masked and prefix language modeling. Specifically, in the first stage, following Izacard et al. [105], the retriever and reader are trained jointly with retrieval-augmented masked language modeling. The training objective for the retriever is to minimize the KL divergence $\mathrm{KL}(p_{\text{READER}} \parallel p_{\text{RETRIEVER}})$ between the passage posterior distribution according to the reader and the passage distribution from the retriever over the top-K retrieved passages, i.e., $p_{\text{READER}}(d) = \frac{\exp(\log p_{LM}(a \mid d,q))}{\sum_{i=1}^{K}\exp(\log p_{LM}(a \mid d_i,q))}$, $p_{\text{RETRIEVER}}(d) = \frac{\exp(s(d,q)/T)}{\sum_{i=1}^{K}\exp(s(d_i,q)/T)}$, where $s(\cdot)$ calculates the dot product between the query q and passage d vectors, and T is a hyperparameter. The training objective for the reader is to maximize the likelihood of the masked spans with n retrieved passages: $\sum_i \log p(a_i \mid q, \{d_k\}_{1,\ldots,n}, a_{1:i-1})$.

In the second stage, for each sequence, we mask 10% of the tokens on average at the end of the sequence with the <extra_id_0> token. Then, we use the retriever obtained from the first stage to retrieve relevant passages using the prefix and train the reader to recover

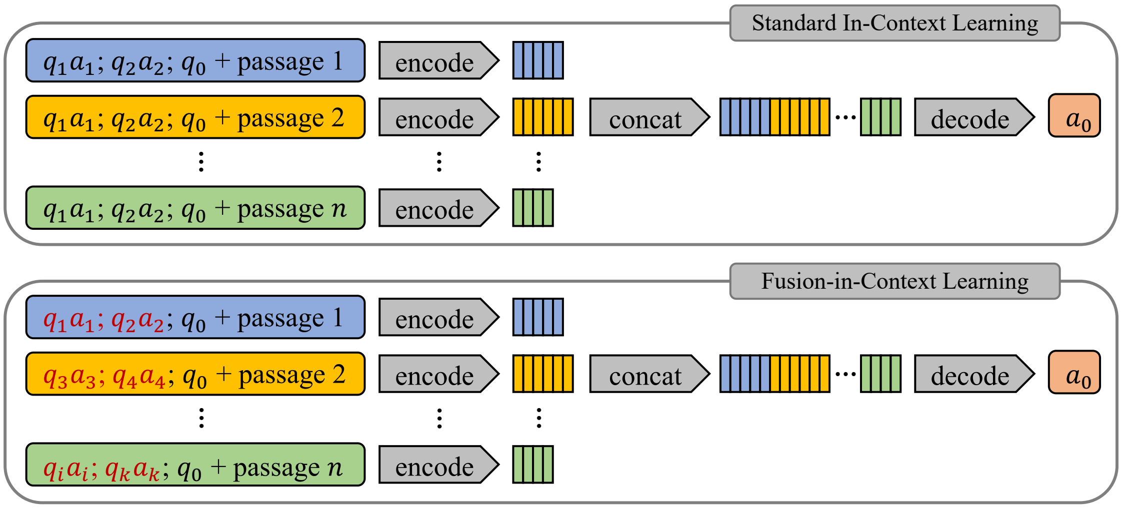


Figure 6.5: Standard In-Context Learning vs Fusion-in-Context Learning.

the suffix of this sequence with the prefix and the passages as input. An example of input and output for retrieval-augmented prefix language modeling is shown in Figure 6.4. We can observe that the pretraining objective aligns more closely with prompting strategy 1 in Figure 6.1. We refer to the model trained with this combined objective as Raven.

Raven benefits from both the retrieval-augmented masked language modeling, which contributes to a better reader and retriever, and retrieval-augmented prefix language modeling, which mitigates the gap between pretraining and inference. This design is *non-trivial*. In Appendix C.3.1, we verify the effectiveness of it by exploring different training strategies.

6.3.3 Fusion-in-Context Learning

In Section 6.3.1, we observe that the performance does not further improve with more in-context examples after 8-shot. One major reason for this is the limited sequence length during the pretraining process, which makes it difficult for the model to handle long sequences during inference. Pretraining models with longer contexts would be a potential solution, but it would significantly increase computation cost. Additionally, the maximum input length is also constrained by the maximum sequence length of the retriever, i.e., Contriever, which is based on BERT [44] and has a maximum length of 512 tokens.

As an alternative, we propose an approach to enable models to learn from more in-context examples without requiring additional training. As described in Section 6.3.1, the reader is based on the Fusion-in-Decoder architecture [104], where multiple passages are retrieved, and each passage, concatenated with the in-context examples and target question, is fed to the encoder separately (Figure 6.5, top). To allow the model to process more in-context examples, we can feed *different* in-context examples to the encoder with each passage (Figure 6.5, bottom). In this way, the model can incorporate more in-context examples during its inference process. We refer to this strategy as *Fusion-in-Context Learning (FiCL)*.

In implementation, for a k-shot setting, such as a 64-shot setting, to effectively utilize the 64 examples, we randomly shuffle these examples and select m (e.g., 5) examples in order as the input for the encoder each time. If all the examples have been used, we shuffle the 64 examples again. We denote the configuration of FiCL as [k-m], which stands for [k-shot, m-fusion].

6.3.4 In-Context Example Retrieval

Liu et al. [150], Rubin et al. [212], Su et al. [230] demonstrate that a well-chosen selection of in-context examples can enhance in-context learning. Building on this insight, we propose utilizing the retriever of RAVEN to retrieve in-context examples. Specifically, we use RAVEN's retriever to build an index during the preparation step, and then, during testing, when the model receives an input, it could efficiently retrieve in-context examples with its retriever.

By integrating RAVEN's retriever in this manner, we aim to: 1) automate in-context learning, which is particularly practical for model owners who have a database of examples. Without this, users would need to manually provide in-context examples; and 2) optimize the selection of in-context examples, thereby improving in-context learning performance.

6.4 EXPERIMENTS

6.4.1 Experimental Setup

Datasets. Following the setup in Section 6.3.1, we first evaluate on two widely-used open-domain question answering datasets: Natural Questions [127] and TriviaQA [112]. Additionally, we conduct a case study on long-form question answering using the ELI5 dataset [57]. Furthermore, we assess the models' language understanding ability using the Massively Multitask Language Understanding (MMLU) benchmark [81]. Detailed information regarding the MMLU evaluation is in Appendix C.2.5. Other evaluation settings are the same as Section C.2.1.

Baselines. Since both RAVEN and ATLAS [105] are trained starting from T5, we choose ATLAS as a primary baseline for comparison. We also compare our model with decoder-only

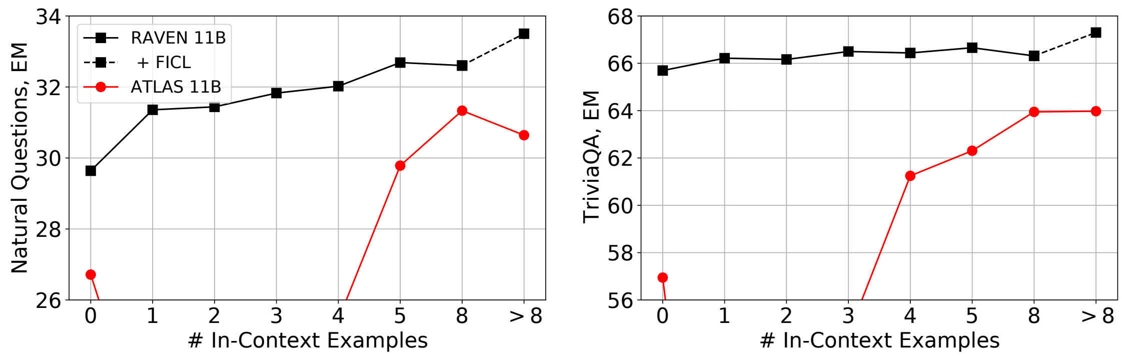


Figure 6.6: Raven vs Atlas. We report the best observed performance achieved with more than eight shots for "> 8".

LLMs such as GPT-3 [24], PaLM [37], and LLaMA [238] (in a closed-book setting). Additionally, for open-domain QA, we evaluate our approach against RePlug [223] and Retro [20], as well as its improved version Retro++ [249]. These models are decoder-only language models augmented with retrieval. RePlug is based on Codex [30] and Contriever [103], where the passages are retrieved by Contriever (using ensemble and additional adaptation) and fed directly to Codex. Retro is a GPT model [205] augmented with a transformer encoder to encode the retrieved passages. Retro++ is a variant of Retro that feeds the most relevant retrieved passage into the GPT decoder while providing other passages to its encoder. For MMLU, we also include FLAN-T5 [38], an enhanced version of T5 that has been trained on a large mixture of tasks with instruction finetuning.[21]

6.4.2 Open-Domain Question Answering

We choose open-domain QA as our primary evaluation task, as it effectively represents knowledge-intensive challenges and is widely employed in real-world applications.

Raven vs Atlas. Figure 6.6 and Table 6.3 present the exact match (EM) scores for Atlas and Raven on the NQ and TriviaQA datasets. Both the 3B and 11B Raven models significantly outperform Atlas. For instance, on TriviaQA, Raven 11B achieves an improvement of 8.8%, 30.7%, and 2.8% in the 0-shot, 1-shot, and few-shot settings respectively, compared to Atlas 11B. Furthermore, the performance of Raven increases steadily with the number of in-context examples, while the performance of Atlas experiences a substantial decline in low-shot settings, demonstrating the effectiveness of Raven across various shot settings.

[21] Implementation details are described in Appendix C.2.2.

		Natural Questions			TriviaQA		
		0-shot	1-shot	few-shot	0-shot	1-shot	few-shot
GPT-3	13B	7.8	13.7	21.0 (64)	41.8	51.3	57.5 (64)
GPT-3	175B	14.6	23.0	29.9 (64)	64.3	68.0	71.2 (64)
PaLM	8B	8.4	10.6	14.6 (5)	39.5	48.5	47.2 (5)
PaLM	62B	18.1	23.1	27.6 (5)	67.3	72.7	70.1 (5)
PaLM	540B	21.2	29.3	39.6 (64)	**76.9**	**81.4**	**81.4** (1)*
Codex	175B	-	-	40.6 (16)	-	-	73.6 (16)
LLaMA	7B	16.8	18.7	26.1 (64)	50.0	53.4	57.6 (64)
LLaMA	65B	23.8	31.0	39.9 (64)	68.2	71.6	73.0 (64)
Retrieval-Augmented Language Models							
Codex + Contriever	175B	-	-	44.2 (16)	-	-	76.0 (16)
Codex + RePlug	175B	-	-	44.7 (16)	-	-	76.8 (16)
Codex + RePlug LSR	175B	-	-	**45.5** (16)	-	-	77.3 (16)
Retro	9.5B	8.9	-	-	36.0	-	-
Retro++	9.5B	25.8	-	-	48.3	-	-
Atlas	3B	23.7	25.1	28.4 (5)	54.3	55.5	61.1 (5)
Atlas + FiCL	3B			29.6 [64-5]			62.0 [64-5]
Atlas	11B	26.7	21.3	31.3 (8)	56.9	35.5	63.9 (8)
Atlas + FiCL	11B			32.0 [64-8]			64.9 [64-8]
Raven	3B	29.3	**31.7**	31.4 (5)	62.4	63.2	62.6 (5)
Raven + FiCL	3B			32.8 [40-1]			63.6 [40-1]
Raven	11B	**29.6**	31.4	32.7 (5)	65.7	66.2	66.7 (5)
Raven + FiCL	11B			33.5 [64-5]			67.3 [64-5]

* For TriviaQA, PaLM's 1-shot performance surpasses other settings. For other models, we select the best k-shot ($k \in \{2, 3, 4, 5, 8, 16\}$) performance or report the number in the original paper.

Table 6.3: Results on NQ and TriviaQA. Since the performance varies significantly depending on the capability of the base model, the results from models other than Atlas should only be used for reference to gauge the position. And we assume Raven can achieve significant performance improvement when based on a stronger base model.

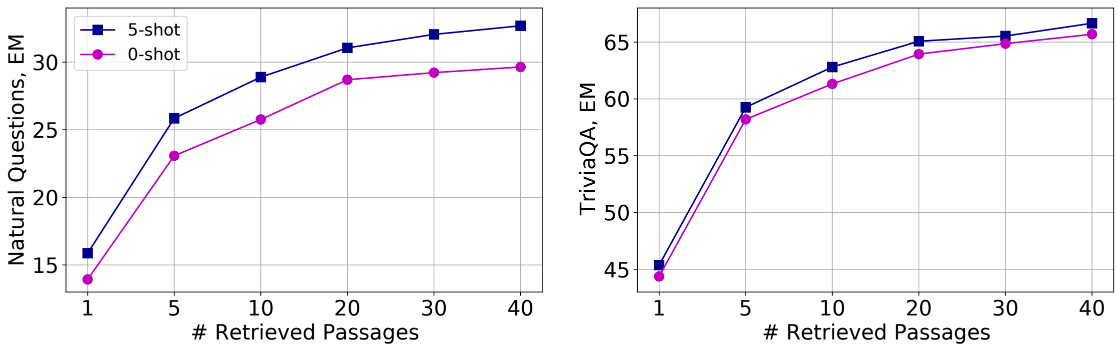


Figure 6.7: Results of RAVEN 11B with different numbers of retrieved passages.

	Natural Questions		**TriviaQA**	
	1-shot	**5-shot**	**1-shot**	**5-shot**
3B	+9.1	+11.6	+0.0	+1.6
11B	+9.8	+11.1	-0.5	+1.0

Table 6.4: Performance improvement of RAVEN with In-Context Example Retrieval.

Fusion-in-Context Learning. We also report the results of models with Fusion-in-Context Learning (FiCL) in Table 6.3. For both ATLAS and RAVEN, FiCL contributes to approximately a 1% improvement, which is not attainable by standard in-context learning, where performance does not further improve (or even decreases) with more than 8 in-context examples. This demonstrates the superiority of FiCL for enabling models to learn from more examples.

Comparison to Other Models. In Table 6.3, we further compare RAVEN to other baselines. On NQ, RAVEN's zero-shot and one-shot performance surpasses all the baselines, including PaLM, even though RAVEN 3B has 180 times fewer parameters than PaLM 540B. The zero-shot performance of RAVEN on TriviaQA is also on par with PaLM 62B. Furthermore, RAVEN's zero-shot performance significantly exceeds that of both RETRO and RETRO++, which are retrieval-augmented language models of a similar scale.

In the few-shot setting, with FiCL, RAVEN achieves performance comparable to GPT-3 175B and PaLM 62B. However, there remains a gap between RAVEN and the larger PaLM 540B and Codex 175B models. Nevertheless, given the considerably smaller scale of RAVEN in comparison to PaLM and Codex, its performance can be considered impressive. The performance of RAVEN may be further improved if it is built upon a larger model, in which case its few-shot performance is likely to surpass that of PaLM and Codex.

Effect of Number of Retrieved Passages. Figure 6.7 illustrates the effect of the number

		0-shot	1-shot	5-shot
GPT-3	13B	-	-	26.0
GPT-3	175B	-	-	43.9
PaLM	8B	-	-	25.3
PaLM	62B	-	-	53.7
PaLM	540B	-	-	69.3
T5	3B	-	-	25.7
T5	11B	-	-	25.9
FLAN-T5	3B	-	-	52.4
FLAN-T5	11B	-	-	55.1
Atlas	3B	43.7	36.9	38.5
+ FiCL	3B			42.6 [40-1]
Atlas	11B	47.4	45.3	44.2
+ FiCL	11B			48.0 [40-1]
Raven	3B	45.7	40.0	40.4
+ FiCL	3B			44.5 [64-5]
Raven	11B	48.9	49.2	48.7
+ FiCL	11B			50.5 [40-1]

Table 6.5: Results on MMLU.

of retrieved passages. As the number of retrieved passages increases, we observe a significant performance improvement of Raven 11B in both the 0-shot and 5-shot settings.

In-Context Example Retrieval. Section 6.3.4 suggests using Raven's retriever for in-context example retrieval. Results in Table 6.4 show that this approach improves Raven's few-shot results, especially on NQ where a ∼10% improvement is observed. This indicates the positive impact of incorporating more relevant in-context examples.

Additional Results. We conduct an **ablation study of different training strategies** in Appendix C.3.1 and provide a **case study on long-form question answering** in Appendix C.3.2.

6.4.3 MMLU

Table 6.5 summarizes the results (accuracy) on Massive Multitask Language Understanding (MMLU). We find that the zero-shot performance of Raven is impressive, surpassing the few-shot performance of GPT-3 175B and being slightly worse than PaLM 62B, despite having a significantly smaller number of parameters. Furthermore, with the same number of parameters, the performance of Raven is far superior to T5. Additionally, even without instruction finetuning, Raven achieves performance comparable to FLAN-T5, a model fine-tuned on a large collection of tasks. We expect further improvement of Raven by applying instruction tuning as well and leave it for future study.

Interestingly, with standard in-context learning, the few-shot performance of Raven is worse than zero-shot, possibly due to the longer questions and answer options in MMLU causing context length issues in the 5-shot setting. Also, in the one-shot setting, since MMLU is a multiple-choice QA task, providing only one example might introduce bias in the model's prediction, favoring a specific option. However, with Fusion-in-Context Learning, the performance improves significantly, leading to better few-shot performance for the 11B model compared to its zero-shot performance, further demonstrating the effectiveness of FiCL.

6.5 CONCLUSION

In this study, we have delved into the in-context learning ability of retrieval-augmented encoder-decoder language models. We commenced with a comprehensive analysis of the models in the literature and subsequently developed our model based on the analysis. Our extensive experimental results demonstrated that our model significantly outperforms previous models and achieves results on par with some of the most advanced language models, even with substantially fewer parameters. These findings highlight the potential of retrieval-augmented encoder-decoder language models in the realm of in-context learning.

CHAPTER 7: DEER: DESCRIPTIVE KNOWLEDGE GRAPH FOR KNOWLEDGE REASONING

In the previous chapter, we develop a technique to augment language models with external knowledge through retrieval. However, to answer more complex questions, e.g., questions requiring multi-hop reasoning, directly retrieving relevant paragraphs from a raw corpus based on the input query may not be sufficient.

To enable the model to perform multi-hop knowledge reasoning, a candidate external knowledge source is a knowledge graph. However, knowledge graphs are often far from complete and require significant manual effort to construct, leading to intrinsic limitations when used to augment language models. In this chapter, we design a new data structure, named *Descriptive Knowledge Graph* (denoted as $\mathbb{DEER}$), to solve the above limitations. In $\mathbb{DEER}$, relationships between entities are represented by free-text relation descriptions. For instance, the relationship between the entities *machine learning* and *algorithm* can be represented as "*Machine learning* explores the study and construction of *algorithms* that can learn from and make predictions on data." To construct $\mathbb{DEER}$, we propose a self-supervised learning method to extract relation descriptions with the analysis of dependency patterns and to generate relation descriptions with a transformer-based relation description synthesizing model, where no human labeling is required. Experiments demonstrate that our system can extract and generate high-quality relation descriptions for explaining entity relationships. We also demonstrate an application of $\mathbb{DEER}$ in the biomedical domain and use it as an external knowledge source to facilitate LLMs in performing knowledge reasoning.[22]

7.1 INTRODUCTION

Relationships exist widely between entities. For example, a person may be related to another person or an institution, and a scientific concept can be connected to another concept. At the same time, relationships between entities can be subtle or complex, e.g., the relationship between *machine learning* and *algorithm*.

To model relationships between entities, researchers usually construct knowledge graphs (KGs) [83, 108], where nodes are entities, e.g., *machine learning*, and edges are relations, e.g., *subclass of* (Figure 7.2). However, KGs usually require a pre-specified set of relation types, and the covered relation types are usually coarse-grained and simple. This indicates existing KGs lack two desired features. The first is ***openness***: for entities with a relationship

[22]The material in this chapter is based on Huang et al. [97]. Code and data are available at `https://github.com/jeffhj/DEER`.

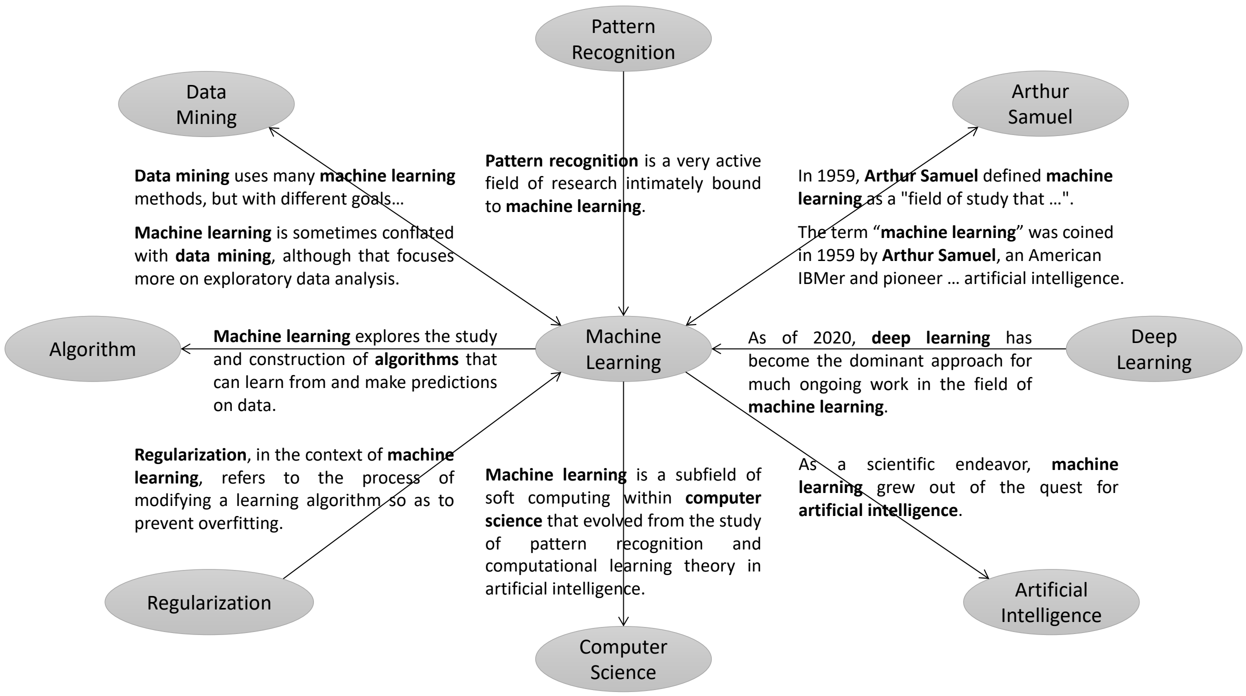


Figure 7.1: Relations in DEER🦌. Here we show *machine learning* and several of its related entities, with corresponding relation descriptions produced by our model (only extraction) in the edges.

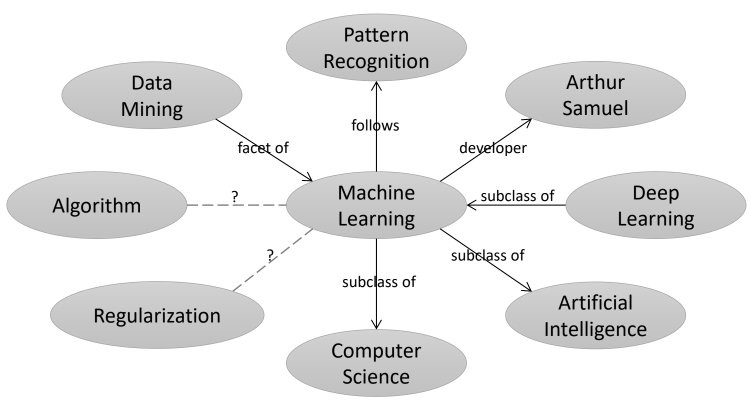


Figure 7.2: Relations in Wikidata (Knowledge Graph), where ? means the relation is not present in the graph.

not covered by the type set, KGs cannot handle their relationship directly. Besides, in many cases, the relationship between entities is complex or idiosyncratic that it cannot be simply categorized to a relation type. For instance, for related entities *machine learning* and *algorithm*, Wikidata [244] does not include a relation for them, and it is also not easy to come up with a relation type to describe their relationship.

The second feature is about ***informativeness***. With the relational facts in KGs, humans may still have difficulty in understanding entity relationships. For instance, from fact "(data mining, *facet of*, database)" in Wikidata, humans may guess *data mining* and *database* are related fields, but they cannot understand how exactly they are related, e.g, *why is it a facet?* and *what is the facet?*

Although techniques like knowledge graph reasoning [33, 130, 264] or open relation extraction [55] can represent more complex relationships to some extent, they do not fundamentally solve the limitations as discussed in Huang et al. [88]. For instance, neither a multi-hop reasoning path in KGs nor a triple extracted by open relation extraction, e.g., (data mining methods, *to be integrate within*, the framework of traditional database systems), is easy to interpret.

Based on the above analysis, we propose a new form of modeling relationships between entities: $\mathbb{DEER}$ (**D**escriptive Knowledge Graph for **E**xplaining **E**ntity **R**elationships). We define $\mathbb{DEER}$ as a graph, where nodes are entities and edges are descriptive statements of entity relationships (refer to Figure 7.1 for an example). $\mathbb{DEER}$ is ***open*** since it does not require a pre-specified set of relation types. In principle, all entity relationships, either explicit or implicit, can be represented by $\mathbb{DEER}$, as long as they can be connected in a sentence—which is not possible for KGs. It is ***informative*** since the relationships between entities are represented by informative free-text relation descriptions, instead of simple short phrases like "facet of".

$\mathbb{DEER}$ has great potential to help users understand entity relationships more easily and intuitively by providing relation descriptions for any two related entities and facilitate downstream tasks on entities and entity relationships such as entity profiling [35, 96, 175], relation extraction [9], and knowledge graph completion [147]. For example, in Figure 7.1, we can understand the semantic meaning of the terms by connecting them with familiar ones. In e-commerce, the system (e.g., Amazon online shopping website) may recommend *tripods* to a photography novice who is browsing *cameras*. An explanation in $\mathbb{DEER}$, e.g., "*tripods* are used for both motion and still photography to prevent *camera* movement and provide stability", could not only help users make a better purchase decision but also justify the recommendation. In KG construction and completion, the relation descriptions can serve as knowledge to improve the performance or as explanations to justify the relations in KGs.

The key to building 𝔻𝔼𝔼ℝ is to acquire high-quality relation descriptions. However, writing or collecting relation descriptions manually requires enormous human efforts and expertise (in our human evaluation in Section 7.6.1, it takes ∼3 minutes to evaluate whether a sentence is a good relation description). Considering this, we propose a novel two-step approach to construct 𝔻𝔼𝔼ℝ with Wikipedia, where no manual annotation is required. Specifically, we first ***extract*** relation descriptions from corpus in a self-supervised manner, where a scoring function is introduced to measure the *explicitness*, i.e., how explicit is the relationship represented by the sentence, and *significance*, i.e., how significant is the relationship represented, with the analysis of dependency patterns. Second, based on the extracted graph, a transformer-based relation description synthesizing model is introduced to ***generate*** relation descriptions for interesting entity pairs whose relation descriptions are not extracted in the first step. This allows 𝔻𝔼𝔼ℝ to handle a large number of entity pairs, including those that do not co-occur in the corpus.

Both quantitative and qualitative experiments demonstrate the effectiveness of our proposed methods. We also conduct case study and error analysis and suggest several promising directions for future work—𝔻𝔼𝔼ℝ not only serves as a valuable application in itself to help understand entity relationships, but also has the potential to serve as a knowledge source to facilitate various tasks on entities and entity relationships.

7.2 RELATED WORK

There are several previous attempts on acquiring entity relation descriptions. For instance, Voskarides et al. [243] study a learning to rank problem of ranking relation descriptions by training a Random Forest classifier with manually annotated data. Subsequently, Huang et al. [98] build a pairwise ranking model based on convolutional neural networks by leveraging query-title pairs derived from clickthrough data of a Web search engine, and Voskarides et al. [242] attempt to generate descriptions for relationship instances in KGs by filling created sentence templates with appropriate entities. However, all these methods are not "open". First, they rely and demand heavily on features of entities and relations. Second, these models only deal with entities with several pre-specified relation types, e.g., 9 in Voskarides et al. [243] and 10 in Voskarides et al. [242], and only explicit relation types, e.g., *isMemberOfMusicGroup*, are covered. Notably, Handler and O'Connor [78] propose to extract relation statements, i.e., natural language expressions that begin with one entity and end with the other entity, from a corpus to describe entity relationships. However, the "*acceptability*" used in their work cannot ensure a good relation description. Moreover, these works do not systematically analyze and define what constitutes a good relational description.

The work most relevant to ours is *Open Relation Modeling* [87, 88], which aims to generate relation descriptions for entity pairs. To achieve this, the authors propose to fine-tune BART [135] to reproduce definitions of entities. Compared to their problem, i.e., text generation, the focus of this study is on graph construction. Besides, their relation descriptions are limited to definitional sentences, which assumes that one entity appears in the other's definition; however, the assumption is not true for many related entities. In addition, their methodology does not incorporate sufficient knowledge about entities and relations for generation.

There are also some other works that can be related. For example, Huang et al. [93], Lin et al. [145], Liu et al. [149, 152] study the problem of generating coherent sentences or explanations containing the given common concepts. Agarwal et al. [2], Dognin et al. [45] study the data-to-text generation [126], which aims to convert facts in KGs into natural language. Gunaratna et al. [73] propose to construct an entity context graph with contexts as random paragraphs containing the target entities to help entity embedding. None of them meets the requirements for high-quality relation descriptions.

7.3 DESCRIPTIVE KNOWLEDGE GRAPH FOR EXPLAINING ENTITY RELATIONSHIPS

$\mathbb{DEER}$🦌 is a graph representing entity relationships with sentence descriptions. Formally, we define $\mathbb{DEER}$ as a directed graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relation description facts. A relation description fact is a triple (x, s, y), where $x, y \in \mathcal{E}$ are the *subject* and *object* of s, respectively. s is a sentence describing the relationship between x and y (Figure 7.1).

To build $\mathbb{DEER}$, the first step is to collect entities and identify related entity pairs, which can be simply achieved by utilizing existing resources, e.g., Wikipedia, and entity relevance analysis, e.g., cosine similarity of entity embeddings in Wikipedia2vec [266]. And then, we need to acquire high-quality relation descriptions for entity pairs. Taking entity pair (*machine learning*, *algorithm*) as an example, a relation description of them can be s_1 in Table 7.1. From the perspective of human understanding, we identify three requirements for a good relation description:

- **Explicitness**: The relationship of the target entities is described explicitly. E.g., in s_1, "*machine learning explores the study and construction of algorithms*" describes the relationship explicitly; while in s_2, the relationship between *machine learning* and *algorithm* is expressed implicitly so that the relationship is difficult to reason.
- **Significance**: The relationship of the target entities is the point of the sentence. In s_1, all

#	Sentence
s_1	*Machine learning* explores the study and construction of *algorithms* that can learn from and make predictions on data.
s_2	*Machine learning* is employed in a range of computing tasks where designing and programming explicit, rule-based *algorithms* is infeasible.
s_3	*Machine learning* includes *algorithms* that are adaptive or have adaptive variants, which usually means that the algorithm parameters are automatically adjusted according to statistics about the optimisation thus far.

Table 7.1: Example sentences containing both *machine learning* and *algorithm*.

the tokens in the sentence are associated with the relationship between *machine learning* and *algorithm*; while in s_3, although the description is explicit, "*which ... far*" mainly characterizes *algorithm*, but not the target entity relationship.

- **Correctness**: The relationship between target entities is described correctly.

There are other requirements to ensure a good relation description, e.g., the sentence is coherent, grammatical, of reasonable length. Compared to the above ones, these requirements are general requirements for any sentence, but not specific to our problem; therefore, we put less emphasis on them.

To acquire relation descriptions that satisfy the above requirements, we propose a novel two-step approach: first extracting relation descriptions from a corpus with the analysis of dependency patterns (Section 7.4), and then generating relation descriptions for interesting entity pairs whose relation descriptions are not extracted in the previous step (Section 7.5).

7.4 RELATION DESCRIPTION EXTRACTION

In this section, we introduce our approach for extracting entity relation descriptions from Wikipedia according to the requirements discussed in Section 7.3.

7.4.1 Preprocessing and Filtering

The goal of preprocessing and filtering is to collect entities and map entity pairs to candidate relation descriptions. To ensure correctness, we use Wikipedia dump as the source corpus[23], which is a high-quality corpus covering a wide range of domains. For each article, we extract the plain text by WikiExtractor[24]. We split the Wikipedia articles into sentences

[23] `https://dumps.wikimedia.org` (enwiki/20210320)
[24] `https://github.com/attardi/wikiextractor`

Dependency label	Description
acl	clausal modifier of noun (adjectival clause)
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
det	determiner
mark	marker
meta	meta modifier
neg	negation modifier
nn	noun compound modifier
nmod	modifier of nominal
npmod	noun phrase as adverbial modifier
nummod	numeric modifier
poss	possession modifier
prep	prepositional modifier
quantmod	modifier of quantifier
relcl	relative clause modifier
appos	appositional modifier
aux	auxiliary
auxpass	auxiliary (passive)
compound	compound
cop	copula
ccomp	clausal complement
xcomp	open clausal complement
expl	expletive
punct	punctuation
nsubj	nominal subject
csubj	clausal subject
csubjpass	clausal subject (passive)
dobj	direct object
iobj	indirect object
obj	object
pobj	object of preposition

Table 7.2: Manually collected modifying dependencies in spaCy.

with the NLTK library[25] and map entity pairs to candidate relation descriptions with the following steps:

Entity collection. We collect Wikipedia page titles (surface form) as our entities. To acquire knowledge and utilize the pre-trained entity embeddings in Wikipedia2Vec [266] in the later steps, we only keep entities that can be recognized by Wikipedia2Vec.

[25]`https://www.nltk.org`

Local mention-entity mapping. Wikipedia2Vec uses hyperlinks to collect a global mention-entity dictionary to map the entity mention to the referent entities, like mapping "apple" to "Apple Inc" or "Apple (food)". In this work, we follow a similar approach to build the mapping. To maintain high accuracy and low ambiguity, we craft the entity mention from the entity by removing the content wrapped by parenthesis and the content after the first comma. For example, a mention-entity pair could be ("Champaign", "Champaign, Illinois") or ("Python", "Python (programming language)"). Unlike Wikipedia2Vec, we create a local dictionary for each Wikipedia page. When processing a page, we dynamically update the dictionary with mention-entity pairs collected from the hyperlinks, and extract the entity occurrence with the updating dictionary in one pass. This can reduce the ambiguity when two entities with the same entity mention co-occur on one page and also avoid collecting trivial entity occurrence on the page.

Hyperlink mapping correction. Using hyperlinks to collect entities will lead to errors under some conditions: 1) The original link is redirected to a new page, where the title does not match with the entity in the link; 2) The entity in the link is lower-cased and thus, does not match with any title. Under the first condition, we just skip this entity because we require that the entity mention must appear in the sentence to prove its occurrence. Under the second situation, if there is only one page title matching with the entity under the case-insensitive setting, we correct the entity to this page title. Otherwise, if there is more than one match, we use the entity embeddings in Wikipedia2Vec to measure the cosine similarity between each matched title and the title of the current page and correct the entity with the most relevant one.

Filtering. Sometimes the entity mention extracted from the sentence may be part of a bigger noun phrase, which is not an entity mention. For example, suppose we recognize "algorithm" and "graph" as entity mentions in the sentence "The breadth-first-search algorithm is a way to explore the vertexes of a graph layer by layer." However, this is not a good relation description between"algorithm" and "graph" because the subject is "breadth-first-search algorithm" rather than "algorithm". Therefore, it is necessary to determine the completed noun phrase for each entity mention. With the dependency tree of the sentence, we recursively find all the child tokens and the ancestor tokens that are connected to the entity mention with a *compound* dependency. To avoid any confusion, we simply reject the entity occurrence if its completed noun phrase and entity mention are different.

Besides, to ensure that the length of relation descriptions is reasonable, we only keep the sentences with the number of tokens $\in [5, 50]$. We also only keep sentences whose shortest

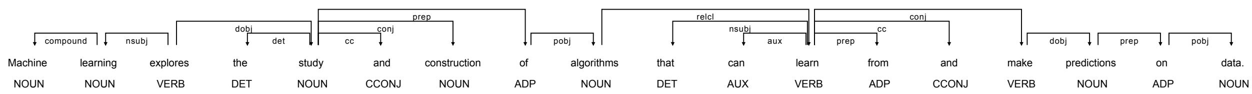


Figure 7.3: Dependency tree of s_1.

dependency path pattern between two target entities starts with *nsubj* or *nsubjpass* (more details are in Section 7.4.2).

7.4.2 Scoring

In this section, we design a scoring function to measure the quality of relation descriptions. Since we use Wikipedia as the source corpus, the *correctness* of the extracted sentences can be largely guaranteed; thus, we focus on measuring *explicitness* and *significance* of candidate relation descriptions.

Shortest Dependency Path as Relation

Inspired by Wu and Weld [263], we use the shortest dependency path to represent the relation pattern between the target entities in a sentence. For instance, Figure 7.3 shows the dependency tree of s_1 processed by spaCy[26]. The shortest path between *machine learning* and *algorithm* is: "learning $\overleftarrow{nsubj}$ explores $\overrightarrow{dobj}$ study $\overrightarrow{prep}$ of $\overrightarrow{pobj}$ algorithms". Following their notation, we call such a path a *corePath*. To represent the relation pattern, we collect dependencies in the path and append "i_" to the dependencies with an inversed direction. E.g., the relation pattern for the above path is $[i\_nsubj, dobj, prep, pobj]$. We remove dependencies that do not affect human understanding. Specifically, we drop the *conj* and *appos* dependencies and replace two consecutive *prep* with one.

Besides *corePath*, we also collect the shortest paths between the *corePath* and the tokens outside the *corePath* to represent the relationships between entity relationships and tokens. For instance, in Figure 7.3, *construction* is a token outside the *corePath* between *machine learning* and *algorithm*. The shortest path between it and the *corePath* is: "study $\overrightarrow{conj}$ construction". We call this kind of path as *subPath*. Similar to *corePath*, we generate the relation pattern from *subPath* and drop the *conj*, *appos* and *compound* dependencies.

[26] `https://spacy.io`

Explicitness

Given two entities and a candidate relation description s, we measure the explicitness by calculating the normalized logarithmic frequency of the relation pattern of the *corePath*:

$$ExpScore(s) = \frac{\log(f_p + 1)}{\log(f_{max} + 1)}, \tag{7.1}$$

where f_{max} is the frequency of the most frequent *corePath* relation pattern and f_p is the frequency of the relation pattern in the present *corePath*. The intuition here is that humans tend to use explicit structure to explain relations. Thus, we assume that a relation description is more explicit if its relation pattern is more frequent. Intuitively, if a relation pattern is unpopular, it is likely that this pattern is either too complicated or contains some rarely used dependencies. Both of these cases may increase the difficulty in reasoning.

Similar to Wu and Weld [263], we only consider patterns that start with *nsubj* or *nsubjpass*, indicating that one of the target entities is the subject of the sentence. This restriction helps increase the explicitness of the selected relation description sentences because if one entity is the subject, the sentence is likely to contain a "argument-predicate-argument" structure connecting the target entities.

Significance

We measure the significance as the proportion of information that is relevant to the entity relationship in a sentence. To measure the relevance of each token in the sentence to the entity relationship, we divide tokens into three categories: 1) *core token* if the token is in the *corePath*; 2) *modifying token* if the token is in a *subPath* that is connected to the *corePath* through a modifying dependency; and 3) *irrelevant token* for the rest tokens. The intuition here is that a sub-dependency tree connected to the *corePath* with a modifying dependency is supposed to modify the relationship. We predefined a set of modifying dependencies in Table 7.2.

We calculate a score for each token in the sentence based on its category and dependency analysis. Then, the significance score is the average of all the token's scores. Formally, for a candidate relation description s, the significance score is

$$SigScore(s) = \frac{\sum_{t \in s} w(t)}{|s|}, \tag{7.2}$$

where

$$w(t) = \begin{cases} 1 & \text{if } t \in ct \\ \frac{\log(f'_{pt}+1)}{\log(f'_{max}+1)} & \text{if } t \in mt \\ 0 & \text{otherwise} \end{cases}, \tag{7.3}$$

where ct is the set of *core tokens* and mt is the set of *modifying tokens*. f'_{pt} is the frequency of the *subPath* relation pattern from the *corePath* to the present token t and f'_{max} is the frequency of the most frequent *subPath* relation pattern. The intuition is: with higher relation pattern frequency, the modifying token is more explicitly related to the entity relationship, and thus, should have a higher score. This also comes with another useful characteristic: the score will decrease token by token as we move along the *subPath* because the frequency of a *subPath* relation pattern cannot be greater than the frequency of its parent. With this characteristic, we can penalize the long modifying *subPath* as it will distract the focus from the entity relationship and is less explicitly related to the relationship.

Relation Descriptive Score

To calculate the explicitness and significance, we need to build a database of relation patterns for both *corePath* and *subPath*. We construct both databases with the candidate relation descriptions and corresponding entity pairs collected from Section 7.4.1 with spaCy. We also require the two target entities in the sentence are related to a certain threshold. Intuitively, if two entities are more related, the sentences containing them are more likely to be relation descriptions; therefore, the extracted *corePath* relation patterns are more likely to indicate entity relationships. We measure the relevance of two entities by calculating the cosine similarity of the entity embeddings in Wikipedia2Vec. We filter out entity pairs (and the associated sentences) with a relevance score < 0.5. This leads to a collection of 7,186,996 *corePaths* and 83,265,285 *subPaths*.

With the databases of relation patterns, we can calculate the explicitness and significance scores for a candidate relation description. The final score, named ***Relation Descriptive Score (RDScore)***, is computed as the harmonic mean:

$$RDScore(s) = 2 \cdot \frac{ExpScore(s) \cdot SigScore(s)}{ExpScore(s) + SigScore(s)}. \tag{7.4}$$

For each entity pair, we calculate *RDScore* for all the candidate relation descriptions and select the candidate with the highest score as the final relation description. To build an initial

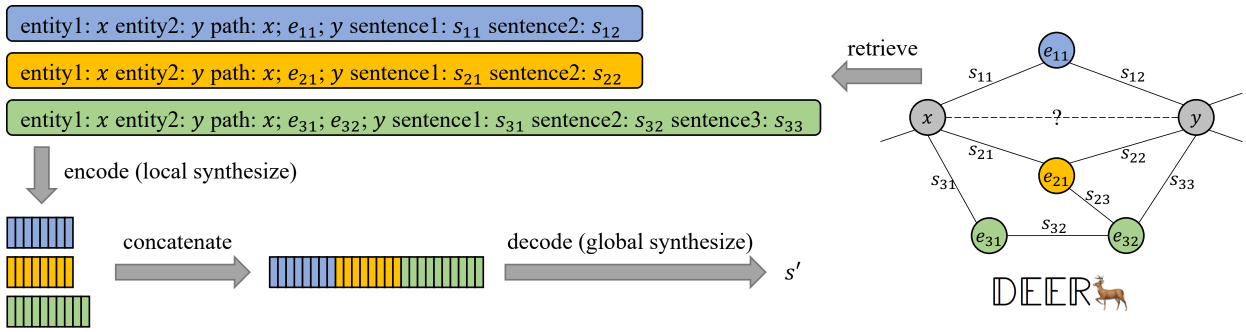


Figure 7.4: The framework of *RelationSyn*. Given entity pair (x, y) whose relation description is not present in the initial $\mathbb{DEER}$, we first retrieve several reasoning paths from the graph. And then, we encode (local synthesize) each reasoning path into a latent vector and concatenate all the latent vectors. Finally, we decode (global synthesize) the vector to produce relation description s' for (x, y).

$\mathbb{DEER}$, we keep edges with an entity relevance score ≥ 0.5[27] and with a relation description whose *RDScore* ≥ 0.75[28]. We refer to this graph as **Wiki-**$\mathbb{DEER}_0$.

7.5 RELATION DESCRIPTION GENERATION

In the previous section, we extract relation descriptions for entity pairs with the analysis of dependency patterns and build an initial $\mathbb{DEER}$ with Wikipedia automatically. However, for some related entity pairs, there may not exist a sentence that contains both entities; and although such a sentence exists, it may not be extracted by the system. To solve this problem, in this section, we introduce *Relation Description Generation*—generating relation descriptions for interesting entity pairs.

We form relation description generation as a conditional text generation task: given two entities, generating a sentence describing the relationship between them with the initial $\mathbb{DEER}$. Formally, we apply the knowledge-enhanced sequence-to-sequence formulation [274]: given an entity pair (x, y) and an initial $\mathbb{DEER}$ $\mathcal{G}_0$, the probability of the output relation description s is computed auto-regressively:

$$P(s|x, y, \mathcal{G}_0) = \prod_{i=1}^{m} P(s_i|s_{0:i-1}, x, y, \mathcal{G}_0), \tag{7.5}$$

[27]Since there is no boundary that delineates whether two entities are related, we consider the relevance threshold as a hyperparameter.

[28]This threshold is also a hyperparameter to balance the density of the graph and the quality of relation descriptions.

where m is the length of s, s_i is the ith token of s, and s_0 is a special start token.

To incorporate $\mathcal{G}_0$ for generation, we propose ***Relation Description Synthesizing (RelationSyn)***. RelationSyn consists of two processes: first retrieving relevant relation descriptions (reasoning paths) from the graph and then synthesizing them into a final relation description (Figure 7.4).

7.5.1 Retrieval

To generate a relation description, the model needs knowledge about the target entities and their relationship. To provide knowledge, we retrieve reasoning paths of the target entities from the graph.

In $\mathbb{DEER}$, we define a reasoning path q as a path connecting the target entities, which is called k-hop if it is connected by k edges. For instance, in Figure 7.4, there are two 2-hop reasoning paths between x and y: $(x, s_{11}, e_{11}, s_{12}, y)$ and $(x, s_{21}, e_{21}, s_{22}, y)$, and two 3-hop reasoning paths: $(x, s_{21}, e_{21}, s_{23}, e_{32}, s_{33}, y)$ and $(x, s_{31}, e_{31}, s_{32}, e_{32}, s_{33}, y)$ in the graph[29]. To measure the quality of reasoning paths, we define *PathScore* as the harmonic mean of *RDScore* of relation descriptions in the path:

$$PathScore(q) = \frac{|\mathcal{S}_q|}{\sum_{s \in \mathcal{S}_q} \frac{1}{RDScore(s)}}, \tag{7.6}$$

where $\mathcal{S}_q$ is the set of relation descriptions in q, and $|\mathcal{S}_q| = k$.

Reasoning paths are helpful for relation description generation. For instance, from reasoning path (deep learning, s'_1, machine learning, s'_2, artificial intelligence) (refer to Figure 7.1 for s'_1 and s'_2), we can infer the relationship between *deep learning* and *AI*: *deep learning* is the dominant approach for *ML*, while *ML* grew out of the quest for *AI*; therefore, *deep learning* is an important technology for the development of *artificial intelligence*.

However, not all reasoning paths are equally useful. Longer reasoning paths are usually more difficult to reason, while paths with higher *PathScore* usually contain more explicit and significant relation descriptions. Therefore, when retrieving reasoning paths for an entity pair, we first sort the paths by their length (shorter first) and then by their *PathScore* (higher first).

[29] In order to collect more reasoning paths as knowledge for generation, we ignore the directions of edges.

# nodes	# edges	average sentence length
1,378,471	2,890,718	19.9

Table 7.3: The statistics of Wiki-$\mathbb{DEER}_0$.

7.5.2 Synthesizing

According to Section 7.5.1, we may retrieve multiple reasoning paths for an entity pair whose relation description is missed in the initial $\mathbb{DEER}$. In this section, we focus on synthesizing relation descriptions in the retrieved reasoning paths into a final relation description of the target entities based on T5 [207] and Fusion-in-Decoder [104].

We first convert each reasoning path to a sequence using the following encoding scheme: e.g., $(x, s_{31}, e_{31}, s_{32},$
$e_{32}, s_{33}, y) \rightarrow$ "entity1: x entity2: y path: x; e_{31}; e_{32}; y sentence1: s_{31} sentence2: s_{32} sentence3: s_{33}". And then, we encode the sequence with the encoder of T5. In this way, the relation descriptions in each reasoning path are synthesized into a latent vector, named "**local synthesizing**".

After local synthesizing, we concatenate the latent vectors of all the retrieved reasoning paths to form a global latent vector. The decoder of T5 performs attention over the global latent vector and produces the final relation description. We name this process as "**global synthesizing**".

Combining retrieval and synthesizing, given two entities, we first retrieve m reasoning paths connecting the target entities according to their length and *PathScore*, and then synthesize them to produce the target relation description. We refer to this model as **RelationSyn-**m.

7.6 EVALUATION

In this section, we verify the proposed methods for building $\mathbb{DEER}$ by conducting experiments on relation description extraction and generation.

7.6.1 Relation Description Extraction

We first present the statistics of the initial $\mathbb{DEER}$ built with Wikipedia in Table 7.3.

To evaluate the quality of relation descriptions in the graph, we randomly sample 100 entity

Rating	Criterion
5	The relation description is explicit, significant, and correct, with which users can understand the relationship correctly and easily.
4	The relation description is a bit less explicit (reasoning is a bit indirect or description is a bit unclear), less significant (containing a little irrelevant content), and less correct (containing minor errors that do not affect the understanding).
3	The relation description is fairly explicit, significant, and correct, while users can still understand the relationship.
2	The relation description is not explicit (reasoning is difficult or description is unclear), significant (containing much irrelevant content), or correct (containing major errors that affect the understanding), while users can still infer the relationship to some extent.
1	The relation description is completely wrong or does not show any relationship between the two entities.

Table 7.4: Annotation guidelines excerpt.

	Rating (1-5)
Random	2.75
ExpScore	3.77
SigScore	3.84
RDScore	**4.18**

Table 7.5: Qualitative results of extraction.

pairs from the graph[30] and ask three human annotators (graduate students doing research on computational linguistics) to assign a graded value (1-5) for each relation description according to Table 7.4.

Since previous works on relation description extraction are supervised and only limited to several explicit relation types, e.g., 9 in Voskarides et al. [243], it is impractical and meaningless to compare with them. For instance, the relationship of (*Arthur Samuel*, *Machine Learning*) is not available or even not considered by the previous methods. Therefore, we verify the effectiveness of our model by comparing different variants of the model:

- **Random**: A sentence containing the target entities is randomly selected as the relation description.
- **ExpScore**: The sentence with the highest *explicitness* is selected according to Eq. (7.1).
- **SigScore**: The sentence with the highest *significance* is selected according to Eq. (7.2).

[30] More specifically, for better comparison with generation later, we sample 100 entity pairs from the test set in Table 7.6.

	train	valid	test
size	847,792	17,662	17,663

Table 7.6: The statistics of data for generation.

- **RDScore**: The sentence with the highest *RDScore* is selected according to Eq. (7.4).

Table 7.5 shows the human evaluation results for relation description extraction, with an average pairwise Cohen's κ of 0.66 (good agreement). From the results, we observe that both our explicitness and significance measurements are important to ensure a good relation description. In addition, *RDScore* achieves an average rating of 4.18, which means that most of the selected sentences are high-quality relation descriptions, further indicating that the quality of Wiki-$\mathbb{DEER}_0$ is high.

7.6.2 Relation Description Generation

Experimental Setup

Data construction. We build a dataset for relation description generation as follows: for an entity pair with a relation description in Wiki-$\mathbb{DEER}_0$, we hide the relation description and consider it as the target for generation. The goal is to recover/generate the target relation description with the rest of the graph[31]. For instance, in Figure 7.4, we hide the edge (relation description s) between x and y and use the remaining reasoning paths to recover s. We train and test on entity pairs with ≥ 5 reasoning paths connecting them. The statistics of the data are reported in Table 7.6.

Models. The task of relation description generation is relevant to *Open Relation Modeling* [88]—a recent work aimed at generating sentences capturing general relations between entities conditioned on entity pairs. To the best of our knowledge, no other existing work can generate relation descriptions for any two related entities (since open relation modeling has only just been introduced). Therefore, we mainly compare the models proposed in Huang et al. [88] with several variants of our model:

- **RelationBART (Vanilla)**: The vanilla model proposed in Huang et al. [88] for generating entity relation descriptions, where BART [135] is fine-tuned on a training data whose inputs are entity pairs and outputs are corresponding relation descriptions.

[31] To increase the difficulty of the task, we assume these two entities do not co-occur in the corpus, i.e., we do not utilize any sentence containing both the target entities for generation.

- **RelationBART-MP + PS**: The best model proposed in Huang et al. [88], which incorporates Wikidata by selecting the most interpretable and informative reasoning path in the KG automatically for helping generate relation descriptions.
- **RelationSyn-**0: A reduced variant of our model, where the encoding scheme of the input is only "entity1: x entity2: y", i.e., no reasoning path and relation description is fed to the encoder.
- **RelationSyn-**m: The proposed relation description synthesizing model (Section 7.5), where m is the maximum number of retrieved reasoning paths for an entity pair.

Metrics. We perform both quantitative and qualitative evaluation. Following Huang et al. [88], we apply several automatic metrics, including BLEU [186], ROUGE-L [146], METEOR [12], and BERTScore [278]. Among them, BLEU, ROUGE, and METEOR focus on measuring surface similarities between the generated relation descriptions and the target relation descriptions, and BERTScore is based on the similarities of contextual token embeddings. We also ask three human annotators to evaluate the output relation descriptions with the same rating scale in Table 7.4.

Implementation details. We train and evaluate all the baselines and variants on the same train/valid/test split. For *RelationBART (Vanilla)* and *RelationBART-MP + PS*, we apply the official implementation[32] and adopt the default hyperparameters. The training converges in 50 epochs. For our models, we modify the implementation of Fusion-in-Decoder[33] and initialize the model with the T5-base configuration. All the baseline models for *RelationSyn* are trained under the same batch size of 8 with a learning rate of 0.0001 and evaluated on the validation set every 5000 steps. The training is considered converged and terminated with no better performance on the validation set in 20 evaluations. The training of all models converges in 20 epochs. The training time is about one week on a single NVIDIA A40 GPU. For evaluation, the signature of BERTScore is: roberta-large-mnli L19 no-idf version=0.3.11(hug trans=4.15.0).

Quantitative Evaluation

Table 7.7 reports the results of relation description generation with the automatic metrics. We observe that our best model *RelationSyn*-5 outperforms the state-of-the-art model for open relation modeling significantly. We also observe that *RelationSyn*-1 performs better

[32] `https://github.com/jeffhj/open-relation-modeling`
[33] `https://github.com/facebookresearch/FiD`

	BLEU	ROUGE	METEOR	BERTScore
RealtionBART-Vanilla [88]	19.61	41.52	20.48	82.99
RealtionBART-MP + PS [88]	21.64	42.62	21.40	83.29
RelationSyn-0	20.83	41.46	20.66	82.84
RelationSyn-1	22.43	42.74	21.65	83.41
RelationSyn-3	23.26	43.33	22.12	83.63
RelationSyn-5	**23.88**	**43.56**	**22.40**	**83.70**

Table 7.7: Quantitative results of relation description generation.

	Rating (1-5)
Random	2.75
RDScore (Oracle)	4.18
RealtionBART-MP + PS	3.12
RelationSyn-0	3.08
RelationSyn-1	3.34
RelationSyn-5	**3.47**

Table 7.8: Qualitative results of generation.

than *RelationSyn*-0, which means that reasoning paths in $\mathbb{DEER}$ are helpful for relation description generation. In addition, as the number of reasoning paths, i.e., m, increases, the performance of RelationSyn-m improves. This demonstrates that the proposed model can synthesize multiple relation descriptions in different reasoning paths into a final relation description.

Qualitative Evaluation

We also conduct qualitative experiments to measure the quality of generated relation descriptions. For a better comparison with extraction, we sample the same 100 entity pairs from the test set as in Section 7.6.1. From the results in Table 7.8, we observe that the quality of generated relation descriptions is higher than that of random sentences containing the target entities. The best model, *RelationSyn*-5, achieves a rating of 3.47, which means the model can generate reasonable relation descriptions. However, the performance is still much worse than *Oracle*, i.e., relation descriptions extracted by our best extraction model (*RDScore*). This indicates that generating high-quality relation descriptions is still a challenging task.

7.6.3 Case Study and Error Analysis

In Table 7.9, we show some sample outputs in the test set of relation description generation of three extraction models: *ExpScore*, *SigScore*, *RDScore*, and three generation models: *RelationSyn*-0, *RelationSyn*-1, *RelationSyn*-5.

For extraction, we observe that if we only consider the explicitness of the sentence, the selected sentence may contain a lot of stuff that is irrelevant to the entity relationship, e.g., (*Mucus*, *Stomach*). And if we only consider the significance, the relationship between entities may be described implicitly; thus the relationship is difficult to reason out, e.g., (*Surfers Paradise*, *Queensland*) and (*Knowledge*, *Epistemology*). And the combination of them, i.e., *RDScore*, yields better relation descriptions.

For generation, we notice that *RelationSyn*-0 suffers severely from *hallucinations*, i.e., generating irrelevant or contradicted facts. E.g., the relation descriptions generated for (*Dayan Khan*, *Oirats*) is incorrect. By incorporating relation descriptions in the reasoning paths as knowledge, hallucination is alleviated to some extent, leading to better performance of *RelationSyn*-1 and *RelationSyn*-5.

From the human evaluation results, we also find that the correctness of relation descriptions extracted by *RDScore* is largely guaranteed. However, sometimes, the extracted sentences are still a bit implicit or not significant. In contrast to this, the relation descriptions generated by *RelationSyn* are usually explicit and significant (the average *RDScore* of the relation descriptions generated by *RelationSyn*-5 is 0.886, compared to 0.853 of *Oracle*), but contain major or minor errors. We think this is because most of the relation descriptions extracted by *RDScore* are explicit and significant, and the generation model can mimic the dominant style of relation descriptions in the training set. However, it is still challenging to generate fully correct relation descriptions by synthesizing existing relation descriptions.

We also attempted to find the eight entity pairs in Table 7.9 in Wikidata. Among them, only (*Surfers Paradise*, *Queensland*) is present in Wikidata. This further confirms that $\mathbb{DEER}$ can model a wider range of entity relationships.

7.6.4 Application

In Figure 7.5, we demonstrate an application of $\mathbb{DEER}$ in the biomedical domain. First, the constructed descriptive knowledge graph itself can serve as the exploration system, facilitating efficient retrieval of relational knowledge and enabling tasks such as drug repurposing and literature curation. Second, the graph can serve as a knowledge source for LLMs such as ChatGPT to summarize information and perform knowledge reasoning between related

	ExpScore	SigScore	RDScore	RelationSyn-0	RelationSyn-1	RelationSyn-5
(Mucus, Stomach)	As the first two chemicals may damage the stomach wall, *mucus* is secreted by the *stomach*, providing a slimy layer that acts as a shield against the damaging effects of the chemicals.	The *mucus* produced by these cells is extremely important, as it prevents the *stomach* from digesting itself.	The *mucus* produced by these cells is extremely important, as it prevents the *stomach* from digesting itself.	*Mucus* is a fluid that is produced by the *stomach*.	*Mucus* is the main barrier to mucus from the *stomach*.	*Mucus* is a thick, protective fluid that is secreted by the *stomach*.
(Surfers Paradise, Queensland)	*Surfers Paradise* is a coastal town and suburb in the City of Gold Coast, *Queensland*, Australia.	In 2009 as part of the Q150 celebrations, *Surfers Paradise* was announced as one of the Q150 Icons of *Queensland* for its role as a "location".	*Surfers Paradise* is a coastal town and suburb in the City of Gold Coast, *Queensland*, Australia.	*Surfers Paradise* is a coastal suburb in the City of Brisbane, *Queensland*, Australia.	*Surfers Paradise* is a coastal town and locality in the Shire of Mareeba, *Queensland*, Australia.	*Surfers Paradise* is a coastal suburb in the City of Redland, *Queensland*, Australia.
(Parkinson's disease, Dopamine)	Thus for the first time the reserpine-induced Parkinsonism in laboratory animals and, by implication, *Parkinson's disease* in humans was related to depletion of striatal *dopamine*.	*Parkinson's disease* is characterized by the death of cells that produce *dopamine*, a neurotransmitter.	*Parkinson's disease* is associated with the degeneration of *dopamine* and other neurodegenerative events.	*Parkinson's disease* is a neurodegenerative disease involving the loss of *dopamine* in the brain.	*Parkinson's disease* is a neurodegenerative disease characterized by the loss of *dopamine*.	*Parkinson's disease* is a neurodegenerative disorder characterized by a slow and steady loss of *dopamine* in the substantia nigra.
(Dayan Khan, Oirats)	Mandukhai and *Dayan Khan* defeated the *Oirats* and the taishis who ruled the Yellow River Mongols.	By 1510 *Dayan Khan* had unified the entire Mongol nation including *Oirats*.	By 1510 *Dayan Khan* had unified the entire Mongol nation including *Oirats*.	*Dayan Khan* was a khan of the *Oirats*.	*Dayan Khan* was a khan of the *Oirats*.	*Dayan Khan* defeated the *Oirats* in 1510 with the assistance of the Four Oirats.
(Knowledge, Epistemology)	In *epistemology*, descriptive *knowledge* is knowledge that can be expressed in a declarative sentence or an indicative proposition.	These questions, but particularly the problem of how experience and *knowledge* interrelate, have broad theoretical and practical implications for such academic disciplines as *epistemology*, linguistics, and psychology.	*Knowledge* is the primary subject of the field of *epistemology*, which studies what we know, how we come to know it, and what it means to know something.	In *epistemology*, *knowledge* is a description of the possible meaning of knowledge.	In philosophy, aristocratic *knowledge* is a form of knowledge that can be gained through experience, through the use of a method of *epistemology*.	In the philosophy of *epistemology*, *knowledge* is often referred to as "a priori" or "synthetic".
(Twilight, Sunset)	*Twilight* is the period of night after *sunset* or before sunrise when the Sun still illuminates the sky when it is below the horizon.	Near the summer solstice, there are less than 8 hours between *sunset* and sunrise, with *twilight* lasting past 10 pm.	*Twilight* is the period of night after *sunset* or before sunrise when the Sun still illuminates the sky when it is below the horizon.	*Twilight* is the period of daylight between sunrise and *sunset* when the Sun is below the horizon.	*Twilight* is the period of darkness when the Sun is below the horizon.	*Twilight* is the period of darkness from *sunset* to sunrise when the Sun is below the horizon.

Table 7.9: Sample of relation descriptions produced by *ExpScore*, *SigScore*, *RDScore*, and *RelationSyn-m*.

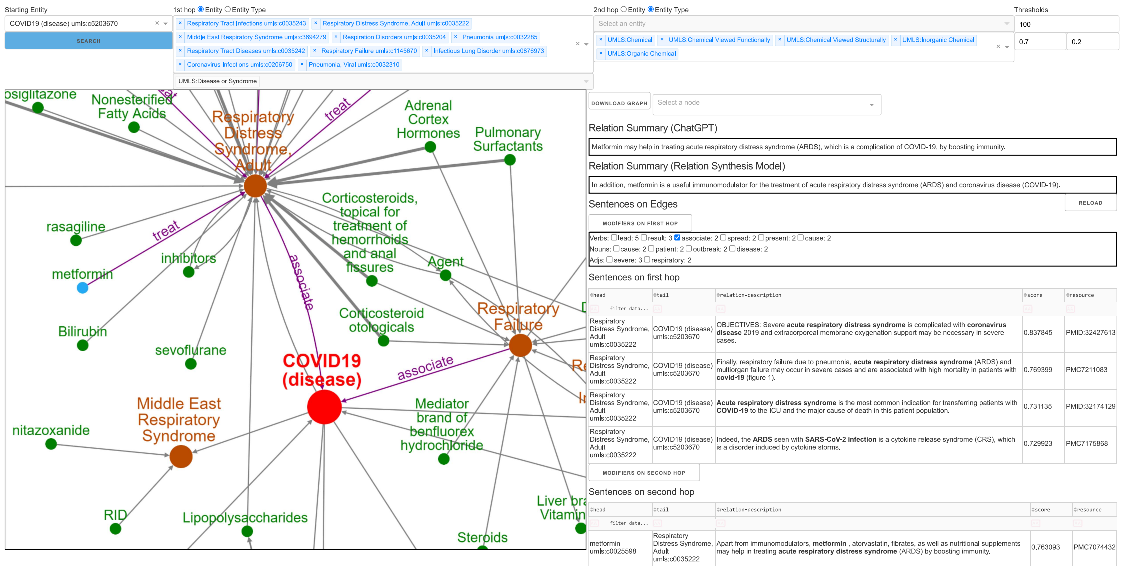


Figure 7.5: The web interface of an exploration system built upon DEER. The interface shows a graph retrieved by a two-hop query: *"COVID19"*—10 *"Disease or Syndrome"* entities—5 *"Pharmacologic Substance"* related entity types. The *metformin* is selected (in blue) and a directed path, *COVID19* → *Respiratory Distress Syndrome, Adult* → *metformin*, is used for relation summary by ChatGPT and relation synthesis model.

entities. We leave the multiple-hop reasoning of LLMs with DEER for more general tasks to future work.[34]

7.7 CONCLUSION

In this chapter, we present DEER🦌—an open and informative form of modeling relationships between entities. To avoid tremendous human efforts, we design a novel self-supervised learning approach to extract relation descriptions from Wikipedia. To provide relation descriptions for related entity pairs whose relation descriptions are not extracted in the previous step, we study relation description generation by synthesizing relation descriptions in the retrieved reasoning paths. We believe that DEER can not only serve as a direct application to help understand entity relationships but also be utilized as an external knowledge source to facilitate LLMs in performing knowledge reasoning.

[34] For more details about the system, please refer to Zhu et al. [286].

CHAPTER 8: CITATION: A KEY TO BUILDING RESPONSIBLE AND ACCOUNTABLE LARGE LANGUAGE MODELS

In this chapter, we explore a novel angle to mitigate risks in Large Language Models (LLMs), drawing parallels between LLMs and established web systems. We identify "citation" as a crucial yet missing component in LLMs, which could enhance content transparency and verifiability while addressing IP and ethical dilemmas. We further propose that a comprehensive citation mechanism for LLMs should account for both non-parametric and parametric content. Despite the complexity of implementing such a citation mechanism, along with the inherent potential pitfalls, we advocate for its development. Building on this foundation, we outline several research problems in this area, aiming to guide future explorations towards building more responsible and accountable LLMs.[35]

8.1 INTRODUCTION

The landscape of artificial intelligence is undergoing rapid transformation, spurred by the emergence of Large Language Models (LLMs) such as ChatGPT/GPT-4 [178, 179]. These models, recognized for their striking ability to generate human-like text, have shown enormous potential in various applications, from information provision to personalized assistance. Nonetheless, their capabilities bring along substantial risks, including intellectual property (IP) and ethical concerns [14, 23, 26, 27, 36, 54, 59, 95, 131, 139, 219].

Research by Carlini et al. [27], Huang et al. [95], for instance, reveals that LLMs are prone to memorizing extensive segments of their training data, including sensitive information. This can result in violations of IP and ethical guidelines. Furthermore, studies by Brown et al. [23], El-Mhamdi et al. [54] suggest that current protective measures fail to provide a comprehensive and meaningful notion of safety for LLMs, making it seemingly impossible to develop safety-preserving, high-accuracy large language models even when trained on public corpora.

While the notion of building an entirely safe LLM might appear daunting, it is crucial to acknowledge that many well-established systems, such as the Web, grapple with similar challenges and have not yet reached absolute safety. Recent legislation like the Online News Act[36], which requires online search engines to compensate Canadian online news outlets for their content, underscores the ongoing issues around content use and compensation on the Web. Furthermore, the Web continues to be a breeding ground for both sensitive information

[35]The material in this chapter is based on Huang and Chang [90].

[36]`https://www.canada.ca/en/canadian-heritage/services/online-news.html`

and misinformation. Hence, expecting a completely risk-free LLM may be an over-ask. Instead, our focus should be on accurately quantifying these risks and developing effective mitigation strategies. It is not about achieving absolute security, but about responsibly managing and minimizing risks in an ethically sound manner.

Guided by these insights, we propose to examine the problem through a different lens: *Can we draw parallels between the risks inherent to LLMs and those experienced by established systems such as search engines and the Web? Can we devise strategies to decrease these risks by aligning with the practices of these mature systems?*

In examining systems like the Web and search engines, we observe a common and robust practice employed to manage IP and ethical concerns: the use of "citations". Broadly defined, a "citation" refers to the act of mentioning or referencing a source or piece of evidence. For example, search engine results also serve as a form of citation, with each entry typically consisting of a title, URL, and brief description. These components collectively cite the webpage's content, offering the user an overview and inviting them to explore the source in greater depth. Citations thus act as anchors for accountability and credit in these systems, providing traceability, preventing plagiarism, and ensuring credit is correctly attributed. They also contribute to transparency, allowing users to verify the information's source.

Upon reflection, it becomes clear that LLMs lack this critical functionality. When LLMs generate content without citations, their output is perceived as independent and self-derived. This creates two significant issues: firstly, when the model produces valuable information, it fails to credit the source it relies on; secondly, when it generates harmful content, it becomes challenging to assign accountability. Incorporating the ability to cite could not only address these ethical and legal conundrums but also bolster the transparency, credibility, and overall integrity of the content generated by LLMs.

However, implementing a "citation" mechanism in LLMs is not as straightforward as it might seem. Unlike the Web, which explicitly links and references sources, LLMs internalize the information and transform it into hidden representations, making accurate citation a significant technical challenge. Although some strides have been made in this direction, as seen in systems like New Bing[37] and Perplexity AI[38], they fall short on several fronts. First, the citations provided in the response of existing systems are often inaccurate [67, 151]. Moreover, these systems typically only cite *non-parametric* content, i.e., content directly retrieved from external sources such as the Web. However, they neglect *parametric* content, the knowledge embedded in the model parameters, which also needs appropriate credit

[37] `https://www.bing.com/new`
[38] `https://www.perplexity.ai`

attribution and consideration for potential harm.

This chapter embarks on an exploratory journey into the potential of integrating a citation mechanism within large language models, examining its prospective benefits, the inherent technical obstacles, and foreseeable pitfalls. We delve into approaches to cite both non-parametric and parametric content, considering the unique characteristics of each type. We also identify and discuss potential setbacks, such as reduced creativity, dissemination of sensitive information, and citation bias. Building on this foundation, we lay bare the hurdles in our path, presenting them as enticing problems for future research. Through this endeavor, we aim to stimulate further discussion and research towards building responsible and accountable large language models.

8.2 OVERVIEW OF LARGE LANGUAGE MODELS

Large language models are typically built on the foundation of transformer architectures [241]. The training process of LLMs usually involves self-supervised learning on vast quantities of text data, including books, articles, and internet content, primarily sourced from the Web. During this stage, models are exposed to diverse textual data, allowing them to learn grammar, facts [197], and even reasoning abilities [89, 258].

Following the initial training, models may undergo further training on smaller, labeled datasets. For instance, ChatGPT [178], a prominent LLM, is fine-tuned on a carefully curated dataset consisting of demonstrations and comparisons, which help the model learn how to generate appropriate responses in conversational contexts.

Risks in LLMs. While LLMs offer numerous benefits, they also pose significant risks [27, 54, 74, 95, 139]. El-Mhamdi et al. [54] highlight these risks, concluding that it is fundamentally impossible to develop safety-preserving, high-accuracy LLMs due to the fundamental intrinsic impossibility of the foundation model learning problem. As they summarized, LLMs achieve optimal performance by employing high-dimensional interpolation, necessitating vast quantities of user-generated data. However, language data from genuine users is intrinsically diverse, with significant variations in individual preferences and styles. This results in empirical heterogeneity, which in turn contributes to the vulnerability of LLMs, particularly when handling sensitive data or encountering fabricated information from fake accounts.

LLMs memorize a lot of training data.	LLMs memorize a lot of training data [1].
Men may be better at STEM.	According to [1], men may be better at STEM.

Figure 8.1: Examples without (left) and with (right) citations. In the first case, citations serve as a way to appropriately credit authors. In the second case, citing the original source of a biased statement ensures that the bias is not misconstrued as the model's opinion.

8.3 "CITATION" IN LLMS

As discussed in the introduction, expecting a risk-free LLM may be an over-ask. The key lies in responsibly managing and minimizing risks in an ethically sound manner. By drawing a comparison between LLMs and the Web, we find that "citation" is a key missing component in LLMs.

Figure 8.1 illustrates model generations with and without citations. In the absence of a citation, there is a potential risk of misunderstanding, leading one to believe that the claim is an opinion or statement formulated by the LLM itself. This not only fails to appropriately credit the original authors, but could also result in ethical dilemmas if the claim is inaccurate or misrepresented.

On the other hand, the inclusion of citations can act as a multifaceted solution to these concerns. Primarily, it helps to mitigate intellectual property and ethical disputes by signaling that the information is not a product of the LLM's "opinion", but a reflection of the cited source. Additionally, citations would enhance the transparency and verifiability of the LLM's output. By indicating the source from which the information is derived, they provide a clear pathway for users to independently verify the validity and context of the information.[39]

8.4 IMPLEMENTATION OF CITATION IN LLMS

In this section, we embark on exploring the potential of incorporating a "citation" mechanism within LLMs. We start our exploration by defining when it would be ideal for an LLM to provide a citation, drawing insights from established practices in academia and ex-

[39] However, citation may also lead to certain potential pitfalls; please refer to Section 8.5 and Section 8.6 for more details.

isting systems like search engines and the Web. We then delve into discussing the possible strategies for effectively implementing citations in LLMs, confronting the methodological and technical intricacies this endeavor involves.

8.4.1 When to Cite?

In academic or professional writing, a citation is typically required when using someone else's ideas, concepts, data, or specific language. For LLMs, determining when to provide a citation is a considerably more challenging task. Given the vast and varied range of queries posed to LLMs, it is crucial to establish when a citation would be appropriate or necessary.

A fundamental rule could be that any fact, idea, or concept that is not general knowledge should be cited. This mirrors the existing conventions on the Web, where sources for specific information are typically provided. For instance, widely known facts like "The Earth revolves around the Sun" would not necessitate a citation, while a less well-known fact like "The fastest spinning stars can rotate more than 600 times per second" would warrant one.

Moreover, the need for a citation could also depend on the nature of the task LLMs are performing. Certain tasks may not necessitate citations, particularly if the output is a reformulation or reinterpretation of the input. For example, in summarization tasks, LLMs condense the input data without introducing new information. The resultant summary is hence an interpretation of the input, and typically, a citation may not be needed for such tasks. Similarly, translation tasks involve converting content from one language to another, without the introduction of novel information.

In essence, while the need for citations in LLMs is task-dependent and context-specific, the guiding principles should be the commitment to knowledge integrity, respect for intellectual property, and adherence to ethical norms. These are similar principles that guide the management of intellectual property and ethical concerns on the Web and in search engines.

8.4.2 How to Cite?

Incorporating citations in LLMs ideally involves connecting outputs to the original source of information. However, this presents a notable technical challenge. During LLMs' training, information is transformed into hidden representations, unlike search engines which possess indices to track and retrieve information. In the case of LLMs, this index is absent, which makes referencing the original source a daunting task. In this section, we delve into the consideration of citations for both *non-parametric* and *parametric* content (Figure 8.2).

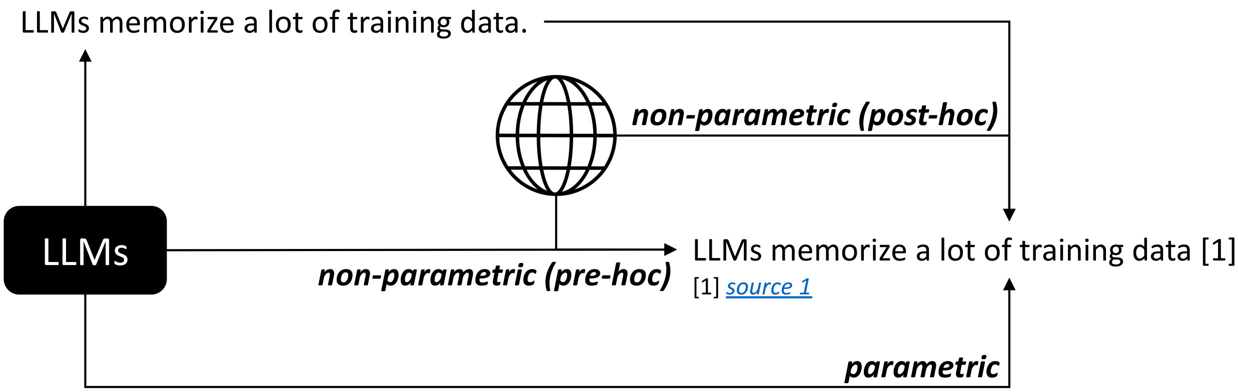


Figure 8.2: *Non-parametric* and *parametric* citations.

Citation for *non-parametric* content

As a potential solution to prevailing challenges, one could design a hybrid system that merges large language models with information retrieval (IR) systems. In this approach, the model is trained to discern when a citation might be required. Subsequently, the IR system is utilized to retrieve relevant sources, namely, *non-parametric* content. The LLM can then incorporate these sources into its responses as citations. We identify two strategies for citing non-parametric content:

- ***Pre-hoc citation***: This approach involves first identifying the need for a citation in the upcoming dialogue or content generation. Once this requirement is recognized, the LLM triggers the IR system to retrieve the necessary information. The LLM then generates its response, seamlessly incorporating the retrieved non-parametric content as citations. This technique can be associated with the broader body of research that augments language models with retrieval [20, 75, 94, 104, 105, 136, 166, 223, 249].
- ***Post-hoc citation***: Conversely, in this strategy, the LLM initially produces a response. An evaluation process then scrutinizes the generated content to ascertain whether a citation is necessary. If a citation is deemed necessary, the IR system is used to locate the appropriate non-parametric content, which is subsequently inserted into the existing text as a citation. Related research includes measuring or requiring attribution in LLMs [65, 67, 84, 151, 209, 275].

In practical applications, a combination of both *pre-hoc* and *post-hoc* citation methods could be adopted for an optimized method. This mixed approach would employ the initial identification and retrieval of potential citations in line with the *pre-hoc* method, followed by a *post-hoc* evaluation to refine the integration of citations based on the generated content. This blend of proactive retrieval and reactive refinement could facilitate the creation of robust, accurate, and well-supported content, while also mitigating intellectual property and ethical concerns surrounding LLMs.

Citation for *parametric* content

In addition to the *non-parametric* content, i.e., content directly retrieved from external sources such as the Web, *parametric* content, which refers to information internalized from the training data, also needs appropriate credit attribution and consideration for potential harm. However, crafting a citation strategy for *parametric* content presents its own set of unique challenges.

The fundamental challenge is the underlying nature of how LLMs process and internalize information. During training, LLMs assimilate vast amounts of data and transform them into an intricate, high-dimensional space that represents learned patterns and structures. The transformation process, rooted in complex mathematical operations, does not inherently retain any clear mapping back to individual data points in the training set. Consequently, generated content cannot easily be traced back to specific training data [10, 72, 124, 187].

This situation is further complicated by the fact that an output generated by LLMs is typically influenced by a multitude of training data points, rather than a single source. This is due to the multi-faceted and context-sensitive nature of language understanding and generation, where a single output can be influenced by a diverse range of linguistic patterns and structures. Thus, the task of accurately attributing a generated output to specific training data pieces is a complex and multifaceted problem that involves unpacking the high-dimensional representations in the model.

Despite these challenges, potential solutions exist. A conceivable approach involves ***training the model with source identifiers***, essentially tags that link specific pieces of information back to their original sources. During training, the model could then be encouraged to retain these identifiers. This would provide a more transparent lineage of information, thereby enhancing accountability. A relevant attempt in this direction was made by Taylor et al. [236], which used special reference tokens to wrap citations and trained models to predict these citations. However, it exhibited certain limitations, such as citation inaccuracy and confinement to academic citations. The successful execution of this method would likely call for advancements in model architecture and training techniques, thereby highlighting intriguing directions for future research.

8.5 PITFALLS OF CITATION IN LLMS

While citations in LLMs can potentially mitigate risks such as IP and ethical issues, as well as improve transparency and verifiability, it is crucial to consider potential pitfalls.

Over-Citation and Sensitive Information Dissemination. The implementation of a citation system in LLMs poses the risk of over-citation, where the excessive use of references might expose more information than necessary. This overexposure could lead to information overload, diluting the significance of critical citations. Moreover, over-citation might inadvertently elevate the risk of disseminating sensitive information [95, 139, 219]. An ill-intentioned user could exploit these extensive citations to gather additional sensitive information.

Inaccurate Citations. Another potential pitfall of implementing citations in LLMs is the risk of inaccurate citations [67, 151]. Given that LLMs may not possess a deep understanding of the content they are trained on or the sources they are citing, there is a chance that they could incorrectly attribute information to a source that does not actually contain that information. Inaccurate citations could mislead users, causing them to believe that a piece of information is verified and supported by a credible source, when in fact, it is not.

Outdated Citations. With the continuous expansion and evolution of knowledge, there is a risk that the sources an LLM cites may become outdated or irrelevant over time. This is particularly likely in fast-evolving fields where new discoveries or advancements quickly supersede existing knowledge. As LLMs are trained on a fixed dataset, their generated content and the sources they cite may not reflect the most current or accurate information. Therefore, there is a potential for LLMs to propagate outdated knowledge, misleading users who rely on the generated content and the cited sources for information.

Propagation of Misinformation. The risk of propagation of misinformation presents a significant concern in the application of LLMs [184]. As LLMs generate output based on the data they have been trained on, there is a chance they could inadvertently cite or echo unreliable or misleading sources, thereby spreading misinformation. This problem could potentially be amplified by the addition of a citation mechanism. A misinterpreted or incorrect citation could be perceived as an authoritative endorsement, inadvertently lending credibility to inaccurate or misleading content.

Citation Bias. Implementing citations in LLMs can also lead to citation bias [14, 71, 107, 167, 218]. Models may tend to cite certain types of sources over others, either due to the

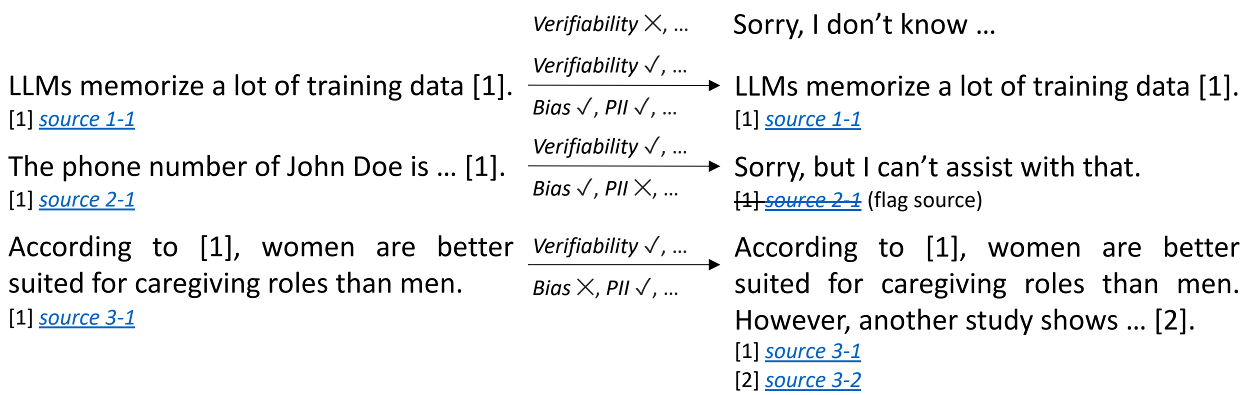


Figure 8.3: Citation with a multifaceted implementation. 1) If a statement cannot be verified by a reliable source, the model can learn to respond with "I don't know"; 2) If the generated output contains sensitive information, such as Personally Identifiable Information (PII), the model should refuse to answer and flag the source to alert the maintainer; 3) If the generated output is detected to reflect a certain bias, the model should refine its response to be more comprehensive and unbiased.

characteristics of the training data or inherent biases in the retrieval mechanism of the IR system. This could lead to an over-reliance on certain types of information and unintentional promotion of certain viewpoints.

Potential for Diminished Creativity. The integration of citations could inadvertently cause a decrease in the creative outputs of the model. When prompted to generate innovative text or propose creative solutions, LLMs might become over-reliant on existing, citable information, thus stifling their novel content generation.

Legal Implications. The utilization of citations could also bring forth legal implications. The introduction of citations could imply a level of responsibility and accountability that the LLM, as an artificial entity, is not equipped to handle. Legal systems around the globe have not yet achieved a consensus on addressing legal issues associated with artificial intelligence, its outputs, and the individuals or entities that create and operate these systems. The inclusion of citations could further complicate these discussions.

8.6 BARRIERS AND RESEARCH PROBLEMS

Building on the potential solutions and pitfalls discussed above, we delve into the primary barriers and corresponding research problems that need to be addressed for successful citation implementation in large language models. Figure 8.3 illustrates examples showing that the inclusion of a citation should be combined with a multifaceted implementation by addressing these research problems.

8.6.1 Determining When to Cite

Deciding when an LLM should cite its sources is a complex issue. While it may be intuitive to suggest that LLMs should always cite sources when they generate information that is not common knowledge (Section 8.4.1), defining what constitutes "common knowledge" is itself a difficult task. Furthermore, as discussed in Section 8.5, it is essential to consider the potential risks associated with over-citation, particularly the increased risk of sensitive information dissemination [95, 139, 219]. LLMs may inadvertently expose sensitive information or contribute to information overload if they include unnecessary or excessive citations. Balancing the need for transparency and accountability with the need to protect privacy and prevent information overload is a critical challenge that needs to be addressed.

8.6.2 Addressing Hallucination in Citation

Hallucination in large language models refers to the phenomenon where the models generate information not grounded in their training data, and that cannot be verified or is simply incorrect [109, 280]. The incorporation of a citation feature can both alleviate and exacerbate this issue. On the one hand, requiring LLMs to link generated information to a tangible source can serve as a form of external verification, potentially restraining the model from generating completely baseless or hallucinated content. The requirement for a source may encourage the model to better align its output with the available data, thereby reducing the likelihood of hallucination.

On the other hand, the citation mechanism itself can potentially hallucinate. If not meticulously designed and implemented, it may end up citing incorrect or non-existent sources [67, 151]. This presents a twofold challenge: Not only is the generated content incorrect, but the citation misleads users into believing that the content is verified and substantiated by the cited source. This issue necessitates the development of techniques to enhance the model's ability to accurately represent the information present in the source, and equip the model to cross-check the consistency of the generated content with the content of the cited source.

8.6.3 Maintaining Temporal Relevance of Citations

In the pursuit of an effective citation mechanism within LLMs, it is essential to address the need for the model's ability to stay updated with the most recent and relevant knowledge.

One potential approach towards this challenge is inspired by the operational principles of search engines. In their bid to stay relevant, search engines continuously update their indexes and ranking algorithms to reflect the latest web. A similar approach could be adopted for LLMs, where they could be designed for ongoing training on updated datasets.

However, executing this in practice presents a significant research problem, considering the scale and complexity of continuously training LLMs and updating their citation mechanisms. Exploring efficient techniques for model training and designing citation mechanisms capable of consistently prioritizing the most recent and relevant sources will require substantial research and development.

8.6.4 Evaluating Source Reliability

Another important challenge is evaluating the reliability of sources used for training data and citations. As mentioned in Section 8.5, LLMs could potentially propagate misinformation if they cite unreliable or misleading sources. While search engines face similar challenges, they are equipped with advanced algorithms to evaluate the reliability and relevance of web pages [181]. Implementing analogous systems within the framework of LLMs presents an interesting and crucial direction for further exploration.

8.6.5 Mitigating Citation Bias

Citation bias in LLMs, as discussed in Section 8.5, can result in the uneven representation of information, leading to the propagation of certain viewpoints while others are neglected. Formulating strategies to curtail such tendencies is paramount.

To begin with, sourcing a more balanced selection of training data can mitigate bias at the inception stage. Ensuring diversity in terms of viewpoints and topics in the training data can reduce bias to some extent.

During citation retrieval, LLMs should utilize an impartial mechanism that does not favor specific types of sources. The underlying algorithms should be optimized to retrieve citations based on their relevance and credibility rather than the prominence of the source or its frequency in the training data.

Finally, the development and application of effective evaluation techniques can help identify and measure any residual bias in LLM outputs. Quantifying the extent of bias enables more targeted corrective measures and provides an objective measure of their efficacy.

8.6.6 Balancing Existing Content with Novel Content Generation

Another intriguing area of research centers on striking a balance between the frequency of citing existing content and generating novel content. LLMs are admired for their capacity to generate creative and unique content [58], as well as their reasoning ability [89, 258]. An over-reliance on citations could potentially inhibit these attributes, reducing the model to a mere aggregator of existing knowledge rather than a generator of new ideas.

Research into this would involve the development of techniques that allow for appropriate citation without hampering the model's creativity. One potential approach could be to create models that are capable of determining the novelty of their generated content and adjusting their citation behavior accordingly. For instance, if a model is generating content based heavily on its training data or the retrieved content, it should provide appropriate citations. Conversely, if the model is generating content that is significantly different from its training data and the retrieved content, it might deem citation unnecessary. Developing such capabilities would require significant advancements in understanding how LLMs generate novel content and how to quantify the 'novelty' of such content.

8.6.7 Navigating Copyright and Fair Use Laws

The application of citation mechanisms in LLMs opens up a new array of legal challenges. Understanding and complying with copyright and fair use laws when citing sources is a complex issue. For instance, how much quoted material from a source would be considered fair use and under what conditions can it be used? In many jurisdictions, the law is not completely clear, especially as it applies to the use of AI technology. Thus, research in the legal aspects of using LLMs for generating text with citations is crucial to ensure legal compliance.

8.7 CONCLUSION

In conclusion, the incorporation of a citation mechanism within LLMs presents a promising approach to numerous challenges, including but not limited to intellectual property rights, ethical concerns, and the need for transparency and verifiability in AI outputs. By equipping LLMs with the ability to accurately attribute the origins of information, we can cultivate a climate of enhanced accountability for the content these models generate. This signifies a progressive step towards constructing a framework of ethical responsibility in AI that respects intellectual property rights and upholds information integrity.

While introducing a citation mechanism in LLMs presents an exciting opportunity for enhancing responsibility and accountability, implementing such a system is not without its technical challenges. We introduce this concept with a hopeful perspective, but readers should be cognizant of the numerous technical hurdles that must be overcome, as highlighted in Section 8.5 and Section 8.6. Additionally, there are other considerations such as system scalability and latency during inference. Nevertheless, these challenges also represent valuable areas for future research and innovation. Through these efforts, we aim to foster more responsible, accountable, and reliable AI systems, ultimately contributing to a better, more trustworthy technological future.

CHAPTER 9: CONCLUSION

This dissertation has embarked on an extensive exploration of the capabilities and risks of Large Language Models (LLMs). Through our comprehensive analyses, we have highlighted the promising potential of LLMs while also acknowledging their significant limitations and challenges. The journey through their reasoning and self-correction ability has unveiled the current state and intrinsic limitations within these models, underscoring the need for ongoing research and development to enhance their capabilities (Chapter 2 & Chapter 3).

Furthermore, our investigation into the ethical implications of LLMs, particularly concerning potential privacy leakage, has shed light on the critical vulnerabilities present in these models. The potential for personal information leakage, despite being somewhat mitigated by the models' limitations in associating specific data points, remains a significant concern that demands attention and action (Chapter 4 & Chapter 5).

In addressing these challenges, we introduced novel methodologies to augment and safeguard LLMs. The development of Raven (Chapter 6) and Descriptive Knowledge Graph (Chapter 7) represents a leap forward in integrating external knowledge to LLMs. Furthermore, our discussion on the importance of citations within LLM outputs illuminates a path toward greater accountability and transparency in AI-generated content (Chapter 8).

In conclusion, the advent of LLMs presents a landscape filled with both extraordinary opportunities and formidable challenges. As we continue to push the boundaries of what these models can achieve, it is imperative that we also remain vigilant and proactive in addressing the ethical considerations that accompany technological advancements. Our hope is that the insights and methodologies presented in this dissertation will contribute to the responsible advancement and application of LLMs, ensuring their positive impact across various domains. The future of artificial intelligence, as envisioned through the lens of this dissertation, should be not only powerful but also principled.

9.1 FUTURE WORK

The trajectory of future research will concentrate on further advancing the capabilities and mitigating the risks of LLMs. This endeavor aims to achieve a higher level of general intelligence within LLMs that is both technologically profound and ethically sound. The forthcoming research areas are outlined below:

Deepening Understanding of Capabilities and Risks of LLMs. Initial efforts have shed light on the capabilities and risks of LLMs, yet these are vast territories with much left to explore. As LLMs continue to evolve, introducing new functionalities and complexities, it becomes crucial to expand the understanding of their capabilities and associated risks. The future work will focus on devising novel methodologies and metrics that more accurately meaure their capabilities and risks, laying the groundwork for the development and evaluation of LLMs.

Enhancing Capabilities of LLMs by Learning from External Feedback. Recognizing the limitations in self-correction and reasoning within LLMs highlights the need for external input and a wider array of data for capability enhancements. Research will delve into enabling LLMs to learn from interactions with their environment and interpret multimodal data, aiming to narrow the gap between current capabilities and a more nuanced, human-like understanding of the world. This involves pioneering new algorithms and methods to facilitate such advanced integration.

Mitigating Risks of LLMs by Alignment and Attribution. Persistent ethical issues in LLMs, despite existing safety measures, indicate the imperative for more robust approaches for reducing risks. The focus will shift towards aligning the generation of LLMs with ethical principles and enhancing transparency through sophisticated attribution mechanisms. The objective is to forge models that not only comprehend and apply a diverse set of ethical guidelines and societal values but also embody mechanisms for clear attribution. This strategy will also involve assessing the impact of LLMs across various societal aspects, including employment, education, and public discourse, supported by empirical research and interdisciplinary partnerships.

This future research path is dedicated to enhancing the capabilities and reducing the risks of LLMs, guiding their development towards beneficial societal outcomes. The goal of these initiatives is to create LLMs that are not only technologically advanced but also ethically responsible, ensuring their positive integration into various domains of human activity. By focusing on these objectives, we aim to contribute to the creation of AI systems that are beneficial for society as a whole.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

[2] Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

[3] Hussam Alkaissi and Samy I McFarlane. 2023. Artificial hallucinations in chatgpt: implications in scientific writing. *Cureus*, 15(2).

[4] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

[5] Cem Anil, Yuhuai Wu, Anders Johan Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Venkatesh Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems*.

[6] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *ArXiv preprint*, abs/2312.11805.

[7] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *ArXiv preprint*, abs/2305.10403.

[8] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. 2022. Large-scale differentially private BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6481–6491, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[9] Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction.

[10] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. 2022. If influence functions are the answer, then what is the question? In *Advances in Neural*

Information Processing Systems, volume 35, pages 17953–17967. Curran Associates, Inc.

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

[12] Satanjeev Banerjee and Alon Lavie. 2005. 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, Ann Arbor, Michigan. Association for Computational Linguistics.

[13] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–718.

[14] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

[15] Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

[16] Prajjwal Bhargava and Vincent Ng. 2022. Commonsense knowledge reasoning and generation with pre-trained language models: A survey. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 12317–12325. AAAI Press.

[17] Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

[18] Sidney Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow.

[19] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *ArXiv preprint*, abs/2108.07258.

[20] 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. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

[21] Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from BERT. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7456–7463. AAAI Press.

[22] Hugo Bronkhorst, Gerrit Roorda, Cor Suhre, and Martin Goedhart. 2020. Logical reasoning in formal and everyday reasoning tasks. *International Journal of Science and Mathematics Education*, 18(8):1673–1694.

[23] Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 2280–2292.

[24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[25] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv preprint*, abs/2303.12712.

[26] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

[27] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6.

[28] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

[29] Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *ArXiv preprint*, abs/2309.13007.

[30] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374.

[31] Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

[32] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *ArXiv preprint*, abs/2211.12588.

[33] Wenhu Chen, Wenhan Xiong, Xifeng Yan, and William Yang Wang. 2018. Variational knowledge graph reasoning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1823–1832, New Orleans, Louisiana. Association for Computational Linguistics.

[34] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *ArXiv preprint*, abs/2304.05128.

[35] Liying Cheng, Dekun Wu, Lidong Bing, Yan Zhang, Zhanming Jie, Wei Lu, and Luo Si. 2020. ENT-DESC: Entity description generation by exploring knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1187–1197, Online. Association for Computational Linguistics.

[36] Simon Chesterman. 2023. Ai-generated content is taking over the world. but who owns it? *But Who Owns it*.

[37] 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. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

[38] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

[39] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv preprint*, abs/1803.05457.

[40] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

[41] Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *ArXiv preprint*, abs/2208.14271.

[42] Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *ArXiv preprint*, abs/2205.09712.

[43] Ishita Dasgupta, Andrew K Lampinen, Stephanie CY Chan, Antonia Creswell, Dharshan Kumaran, James L McClelland, and Felix Hill. 2022. Language models show human-like content effects on reasoning. *ArXiv preprint*, abs/2207.07051.

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

[45] Pierre Dognin, Igor Melnyk, Inkit Padhi, Cicero Nogueira dos Santos, and Payel Das. 2020. DualTKB: A Dual Learning Bridge between Text and Knowledge Base. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8605–8616, Online. Association for Computational Linguistics.

[46] David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-Dickstein, et al. 2022. Language model cascades. *ArXiv preprint*, abs/2207.10342.

[47] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2023. A survey for in-context learning. *ArXiv preprint*, abs/2301.00234.

[48] Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models. *ArXiv preprint*, abs/2209.15003.

[49] Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *ArXiv preprint*, abs/2305.14325.

[50] Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[51] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer.

[52] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

[53] Ward Edwards. 1954. The theory of decision making. *Psychological bulletin*, 51(4):380.

[54] El-Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Lê-Nguyên Hoang, Rafael Pinot, and John Stephan. 2022. On the impossible safety of large ai models. *ArXiv preprint*, abs/2209.15259.

[55] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

[56] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. 2004. *Reasoning about knowledge*. MIT press.

[57] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

[58] Giorgio Franceschelli and Mirco Musolesi. 2023. On the creativity of large language models. *ArXiv preprint*, abs/2304.00008.

[59] Brian L Frye. 2022. Should using an ai text generator to produce academic writing be plagiarism? *Fordham Intellectual Property, Media & Entertainment Law Journal, Forthcoming*.

[60] Yao Fu, Hao Peng, and Tushar Khot. 2022. How does gpt obtain its ability? tracing emergent abilities of language models to their sources.

[61] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *ArXiv preprint*, abs/2210.00720.

[62] Kathleen M Galotti. 1989. Approaches to studying formal and everyday reasoning. *Psychological bulletin*, 105(3):331.

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

[64] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2021. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv preprint*, abs/2101.00027.

[65] Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. 2023. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508.

[66] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. Pal: Program-aided language models. *ArXiv preprint*, abs/2211.10435.

[67] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. *ArXiv preprint*, abs/2305.14627.

[68] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

[69] Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2022. Roscoe: A suite of metrics for scoring step-by-step reasoning. *ArXiv preprint*, abs/2212.07919.

[70] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *ArXiv preprint*, abs/2305.11738.

[71] Steven A Greenberg. 2009. How citation distortions create unfounded authority: analysis of a citation network. *Bmj*, 339.

[72] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. 2023. Studying large language model generalization with influence functions. *ArXiv preprint*, abs/2308.03296.

[73] Kalpa Gunaratna, Yu Wang, and Hongxia Jin. 2021. Entity context graph: Learning entity representations fromsemi-structured textual sources on the web. *ArXiv preprint*, abs/2103.15950.

[74] Shangwei Guo, Chunlong Xie, Jiwei Li, Lingjuan Lyu, and Tianwei Zhang. 2022. Threats to pre-trained language models: Survey and taxonomy. *ArXiv preprint*, abs/2202.06862.

[75] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

[76] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, et al. 2022. Folio: Natural language reasoning with first-order logic. *ArXiv preprint*, abs/2209.00840.

[77] Simon Jerome Han, Keith Ransom, Andrew Perfors, and Charles Kemp. 2022. Human-like property induction is a challenge for large language models.

[78] Abram Handler and Brendan O'Connor. 2018. Relational summarization for corpus analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1760–1769, New Orleans, Louisiana. Association for Computational Linguistics.

[79] Hangfeng He, Hongming Zhang, and Dan Roth. 2023. Rethinking with retrieval: Faithful large language model inference. *ArXiv preprint*, abs/2301.00303.

[80] Chadi Helwe, Chloé Clavel, and Fabian M Suchanek. 2021. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *3rd Conference on Automated Knowledge Base Construction*.

[81] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[82] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

[83] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *Synthesis Lectures on Data, Semantics, and Knowledge*, 12(2):1–257.

[84] Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. 2022. TRUE: Re-evaluating factual consistency evaluation. In *Proceedings of the Second DialDoc Workshop on Document-grounded Dialogue and Conversational Question Answering*, pages 161–175, Dublin, Ireland. Association for Computational Linguistics.

[85] Shlomo Hoory, Amir Feder, Avichai Tendler, Sofia Erell, Alon Peled-Cohen, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, and Yossi Matias. 2021. Learning and evaluating a differentially private pre-trained language model. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1178–1189, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[86] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *ArXiv preprint*, abs/2210.11610.

[87] Jie Huang and Kevin Chang. 2023. VER: Unifying verbalizing entities and relations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15700–15710, Singapore. Association for Computational Linguistics.

[88] Jie Huang, Kevin Chang, Jinjun Xiong, and Wen-mei Hwu. 2022. Open relation modeling: Learning to define relations between entities. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 297–308, Dublin, Ireland. Association for Computational Linguistics.

[89] Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics.

[90] Jie Huang and Kevin Chen-Chuan Chang. 2024. Citation: A key to building responsible and accountable large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*.

[91] Jie Huang, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2023. Can language models be specific? how? In *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics.

[92] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

[93] Jie Huang, Yifan Gao, Zheng Li, Jingfeng Yang, Yangqiu Song, Chao Zhang, Zining Zhu, Haoming Jiang, Kevin Chen-Chuan Chang, and Bing Yin. 2023. CCGen: Explainable complementary concept generation in e-commerce. *ArXiv preprint*, abs/2305.11480.

[94] Jie Huang, Wei Ping, Peng Xu, Mohammad Shoeybi, Kevin Chen-Chuan Chang, and Bryan Catanzaro. 2023. Raven: In-context learning with retrieval-augmented encoder-decoder language models. *ArXiv preprint*, abs/2308.07922.

[95] Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2038–2047, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[96] Jie Huang, Hanyin Shao, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022. Understanding jargon: Combining extraction and generation for definition modeling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3994–4004, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[97] Jie Huang, Kerui Zhu, Kevin Chen-Chuan Chang, Jinjun Xiong, and Wen-mei Hwu. 2022. DEER: Descriptive knowledge graph for explaining entity relationships. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6686–6698, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[98] Jizhou Huang, Wei Zhang, Shiqi Zhao, Shiqiang Ding, and Haifeng Wang. 2017. Learning to explain entity relationships by pairwise ranking with convolutional neural networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4018–4025. ijcai.org.

[99] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *ArXiv preprint*, abs/1904.05342.

[100] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR.

[101] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022. Inner monologue: Embodied reasoning through planning with language models. In *2022 Conference on Robot Learning*.

[102] Michael Huth and Mark Ryan. 2004. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press.

[103] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

[104] Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

[105] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

[106] Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022. TemporalWiki: A lifelong benchmark for training and evaluating ever-evolving language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6237–6250, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[107] Anne-Sophie Jannot, Thomas Agoritsas, Angèle Gayet-Ageron, and Thomas V Perneger. 2013. Citation bias favoring statistically significant studies was present in medical research. *Journal of clinical epidemiology*, 66(3):296–301.

[108] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

[109] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

[110] Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020. X-FACTR: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959, Online. Association for Computational Linguistics.

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

[112] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

[113] Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1279, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[114] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2022. Large language models struggle to learn long-tail knowledge. *ArXiv preprint*, abs/2211.08411.

[115] Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10697–10707. PMLR.

[116] Seyed Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2022. Lambada: Backward chaining for automated reasoning in natural language. *ArXiv preprint*, abs/2212.13894.

[117] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[118] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[119] Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *ArXiv preprint*, abs/2212.14024.

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

[121] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*.

[122] Bennett Kleinberg, Toby Davies, and Maximilian Mozes. 2022. Textwash–automated open-source text anonymisation. *ArXiv preprint*, abs/2208.13081.

[123] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *Proceedings of the 15th European Conference on Machine Learning*, ECML'04, page 217–226, Berlin, Heidelberg. Springer-Verlag.

[124] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

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

[126] Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

[127] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

[128] Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

[129] Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. 2022. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[130] Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK. Association for Computational Linguistics.

[131] Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. 2023. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, pages 3637–3647.

[132] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

[133] Eric Lehman, Sarthak Jain, Karl Pichotta, Yoav Goldberg, and Byron Wallace. 2021. Does BERT pretrained on clinical notes reveal sensitive data? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 946–959, Online. Association for Computational Linguistics.

[134] Edward H Levi. 2013. *An introduction to legal reasoning*. University of Chicago Press.

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

[136] Patrick S. H. 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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[137] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models. *ArXiv preprint*, abs/2206.14858.

[138] Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

[139] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. 2023. Multi-step jailbreaking privacy attacks on chatgpt. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4138–4153.

[140] Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. 2022. Explanations from large language models make small reasoners better. *ArXiv preprint*, abs/2210.06726.

[141] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. 2022. Large language models can be strong differentially private learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[142] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the advance of making language models better reasoners. *ArXiv preprint*, abs/2206.02336.

[143] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *ArXiv preprint*, abs/2305.19118.

[144] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *ArXiv preprint*, abs/2305.20050.

[145] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

[146] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[147] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press.

[148] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

[149] Chenzhengyi Liu, Jie Huang, Kerui Zhu, and Kevin Chen-Chuan Chang. 2023. DimonGen: Diversified generative commonsense reasoning for explaining concept relationships. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4719–4731, Toronto, Canada. Association for Computational Linguistics.

[150] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

[151] Nelson F Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines. *ArXiv preprint*, abs/2304.09848.

[152] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021. KG-BART: knowledge graph-augmented BART for generative commonsense reasoning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6418–6425. AAAI Press.

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

[154] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. *ArXiv preprint*, abs/2301.13688.

[155] Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[156] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*.

[157] Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing leakage of personally identifiable information in language models.

[158] Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith. 2022. Time waits for no one! analysis and challenges of temporal misalignment. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5944–5958, Seattle, United States. Association for Computational Linguistics.

[159] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*.

[160] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[161] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *ArXiv preprint*, abs/2212.08410.

[162] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *ArXiv preprint*, abs/2212.10511.

[163] Christopher D Manning. 2022. Human language understanding & reasoning. *Daedalus*, 151(2):127–138.

[164] Gary Marcus. 2020. The next decade in ai: four steps towards robust artificial intelligence. *ArXiv preprint*, abs/2002.06177.

[165] Conor McHugh and Jonathan Way. 2018. What is reasoning? *Mind*, 127(505):167–196.

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

[167] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. In *Acm sigir forum*, pages 1–27. ACM New York, NY, USA.

[168] Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

[169] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. 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, Florence, Italy. Association for Computational Linguistics.

[170] Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. 2022. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1816–1826, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[171] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

[172] Kanishka Misra, Julia Taylor Rayz, and Allyson Ettinger. 2022. A property induction framework for neural language models. *ArXiv preprint*, abs/2205.06910.

[173] Melanie Mitchell. 2021. Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 1505(1):79–101.

[174] Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

[175] Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 3259–3266. AAAI Press.

[176] 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. 2022. Show your work: Scratchpads for intermediate computation with language models. In *Deep Learning for Code Workshop*.

[177] Theo X Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2023. Demystifying gpt self-repair for code generation. *ArXiv preprint*, abs/2306.09896.

[178] OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. *OpenAI*.

[179] OpenAI. 2023. Gpt-4 technical report.

[180] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

[181] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[182] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *ArXiv preprint*, abs/2308.03188.

[183] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331. IEEE.

[184] Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Yang Wang. 2023. On the risk of misinformation pollution with large language models. *ArXiv preprint*, abs/2305.13661.

[185] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[186] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. 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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[187] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. *ArXiv preprint*, abs/2303.14186.

[188] John Arthur Passmore. 1961. Philosophical reasoning.

[189] Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

[190] Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. 2023. Bidirectional language models are also few-shot learners. In *The Eleventh International Conference on Learning Representations*.

[191] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

[192] Constantinos Patsakis and Nikolaos Lykousas. 2023. Man vs the machine: The struggle for effective text anonymisation in the age of large language models. *ArXiv preprint*, abs/2303.12429.

[193] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *ArXiv preprint*, abs/2304.01904.

[194] Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880, Online. Association for Computational Linguistics.

[195] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

[196] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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

[198] Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Qiang Fu, Yan Gao, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning like program executors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 761–779, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[199] Steven T Piantasodi and Felix Hill. 2022. Meaning without reference in large language models. *ArXiv preprint*, abs/2208.02957.

[200] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *ArXiv preprint*, abs/2210.03350.

[201] Ben Prystawski, Paul Thibodeau, and Noah Goodman. 2022. Psychologically-informed chain-of-thought prompts for metaphor understanding in large language models. *ArXiv preprint*, abs/2209.08141.

[202] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *ArXiv preprint*, abs/2302.06476.

[203] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897.

[204] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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

[206] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv preprint*, abs/2112.11446.

[207] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

[208] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

[209] Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. 2023. Measuring attribution in natural language generation models. *Computational Linguistics*, pages 1–66.

[210] Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot reasoning. *ArXiv preprint*, abs/2202.07206.

[211] Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

[212] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

[213] Laura Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. 2022. Large language models are not zero-shot communicators. *ArXiv preprint*, abs/2210.14986.

[214] Jacob Russin, Randall C O'Reilly, and Yoshua Bengio. 2020. Deep learning needs a prefrontal cortex. *Work Bridging AI Cogn Sci*, 107:603–616.

[215] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. 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 V. 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 Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[216] Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *ArXiv preprint*, abs/2210.01240.

[217] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100.

[218] Chirag Shah and Emily M. Bender. 2022. Situating search. In *Proceedings of the 2022 Conference on Human Information Interaction and Retrieval*, CHIIR '22, page 221–232, New York, NY, USA. Association for Computing Machinery.

[219] Hanyin Shao, Jie Huang, Shen Zheng, and Kevin Chang. 2024. Quantifying association capabilities of large language models and its implications on privacy leakage. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 814–825, St. Julian's, Malta. Association for Computational Linguistics.

[220] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

[221] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *ArXiv preprint*, abs/2210.03057.

[222] Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[223] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *ArXiv preprint*, abs/2301.12652.

[224] Weiyan Shi, Aiqi Cui, Evan Li, Ruoxi Jia, and Zhou Yu. 2022. Selective differential privacy for language modeling. In *Proceedings of the 2022 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2848–2859, Seattle, United States. Association for Computational Linguistics.

[225] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*.

[226] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *ArXiv preprint*, abs/2212.00193.

[227] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *ArXiv preprint*, abs/2201.11990.

[228] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *ArXiv preprint*, abs/2212.04088.

[229] 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. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *ArXiv preprint*, abs/2206.04615.

[230] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Selective annotation makes language models better few-shot learners. In *The Eleventh International Conference on Learning Representations*.

[231] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *ArXiv preprint*, abs/2210.09261.

[232] Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

[233] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

[234] Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[235] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*.

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

[237] Om Dipakbhai Thakkar, Swaroop Ramaswamy, Rajiv Mathews, and Francoise Beaufays. 2021. Understanding unintended memorization in language models under federated learning. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 1–10, Online. Association for Computational Linguistics.

[238] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

[239] Thomas Vakili and Hercules Dalianis. 2021. Are clinical bert models privacy preserving? the difficulty of extracting patient-condition associations.

[240] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

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

[242] Nikos Voskarides, Edgar Meij, and Maarten de Rijke. 2017. Generating descriptions of entity relationships. In *European Conference on Information Retrieval*, pages 317–330. Springer.

[243] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*

(Volume 1: Long Papers), pages 564–574, Beijing, China. Association for Computational Linguistics.

[244] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

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

[246] Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2714–2730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[247] Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2022. Towards understanding chain-of-thought prompting: An empirical study of what matters. *ArXiv preprint*, abs/2212.10001.

[248] Boshi Wang, Xiang Yue, and Huan Sun. 2023. Can chatgpt defend its belief in truth? evaluating llm reasoning via debate. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11865–11881.

[249] Boxin Wang, Wei Ping, Peng Xu, Lawrence McAfee, Zihan Liu, Mohammad Shoeybi, Yi Dong, Oleksii Kuchaiev, Bo Li, Chaowei Xiao, et al. 2023. Shall we pretrain autoregressive language models with retrieval? a comprehensive study. *ArXiv preprint*, abs/2304.06762.

[250] Chenguang Wang, Xiao Liu, and Dawn Song. 2020. Language models are open knowledge graphs. *ArXiv preprint*, abs/2010.11967.

[251] Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A critic for language model generation. *ArXiv preprint*, abs/2308.04592.

[252] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

[253] Peter C Wason. 1968. Reasoning about a rule. *Quarterly journal of experimental psychology*, 20(3):273–281.

[254] Peter Cathcart Wason and Philip Nicholas Johnson-Laird. 1972. *Psychology of reasoning: Structure and content*, volume 86. Harvard University Press.

[255] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does llm safety training fail? *ArXiv preprint*, abs/2307.02483.

[256] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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

[258] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

[259] Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V. Le. 2024. Long-form factuality in large language models.

[260] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*.

[261] Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *ArXiv preprint*, abs/2212.09561.

[262] Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. 2022. Reframing human-AI collaboration for generating free-text explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 632–658, Seattle, United States. Association for Computational Linguistics.

[263] Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden. Association for Computational Linguistics.

[264] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, Copenhagen, Denmark. Association for Computational Linguistics.

[265] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

[266] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30, Online. Association for Computational Linguistics.

[267] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *ArXiv preprint*, abs/2309.03409.

[268] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

[269] Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2022. Language models as inductive reasoners. *ArXiv preprint*, abs/2212.10923.

[270] Qinyuan Ye, Iz Beltagy, Matthew E Peters, Xiang Ren, and Hannaneh Hajishirzi. 2023. Fid-icl: A fusion-in-decoder approach for efficient in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8158–8185.

[271] Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*.

[272] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. 2022. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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

[274] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A survey of knowledge-enhanced text generation. *ArXiv preprint*, abs/2010.04389.

[275] Xiang Yue, Boshi Wang, Kai Zhang, Ziru Chen, Yu Su, and Huan Sun. 2023. Automatic evaluation of attribution by large language models. *ArXiv preprint*, abs/2305.06311.

[276] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

[277] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *ArXiv preprint*, abs/2205.01068.

[278] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[279] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

[280] Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Why does ChatGPT fall short in providing truthful answers? In *I Can't Believe It's Not Better Workshop: Failure Modes in the Age of Foundation Models*.

[281] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2024. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *The Twelfth International Conference on Learning Representations*.

[282] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

[283] Fan Zhou, Haoyu Dong, Qian Liu, Zhoujun Cheng, Shi Han, and Dongmei Zhang. 2022. Reflection of thought: Inversely eliciting numerical reasoning in language models via solving linear systems. *ArXiv preprint*, abs/2210.05075.

[284] Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. 2022. Teaching algorithmic reasoning via in-context learning. *ArXiv preprint*, abs/2211.09066.

[285] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

[286] Kerui Zhu, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Descriptive knowledge graph in biomedical domain. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

[287] Corinne Zimmerman. 2000. The development of scientific reasoning skills. *Developmental review*, 20(1):99–149.

[288] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *ArXiv preprint*, abs/2307.15043.

APPENDIX A: LARGE LANGUAGE MODELS CANNOT SELF-CORRECT REASONING YET

A.1 PROMPTS AND EXAMPLE OUTPUTS

```
Can you solve the following math problem? Christina is planning a birthday party and
needs .75 gift bags per invited guest, because 1/4 of attendees don't show up. She
invited 16 friends. Gift bags are $2 each. How much will she spend? Explain your
reasoning. Your final answer should be a single numerical number, in the form
\boxed{answer}, at the end of your response.

If Christina invited 16 friends, then 1/4 of them won't show up, which means there will
be 16 * 1/4 = 4 friends who won't attend the party. To determine how many gift bags to
buy, we need to subtract the number of friends who won't show up from the total number of
invited guests. The number of invited guests is 16 - 4 = 12. Since Christina needs 0.75
gift bags per invited guest, we can multiply this quantity by the number of invited
guests to find the total number of gift bags needed, which is 0.75 * 12 = 9 gift bags. If
each gift bag is $2, then Christina will spend a total of 2 * 9 = $<<2*9=18>>18 on gift
bags. Therefore, Christina will spend $18 on gift bags. Answer: \boxed{18}. (Incorrect)

Review your previous answer and find problems with your answer.

Upon reviewing my previous answer, I recognize a mistake. I incorrectly stated that
Christina needs 0.75 gift bags per invited guest. The given information states that 1/4
of the attendees won't show up, so the correct calculation should take this into account.
To calculate the number of gift bags needed, we will multiply the number of invited
guests by the ratio of attendees who will actually show up (75%). The number of invited
guests is 16, and if 1/4 of them won't show up, then the ratio of attendees who will show
up is 1 - 1/4 = 3/4. Multiplying the number of invited guests by the ratio of attendees
who will show up gives us: 16 * (3/4) = 12 gift bags needed. Since each gift bag costs
$2, the total amount Christina will spend on gift bags is: 12 * $2 = $<<12*2=24>>24.

Based on the problems you found, improve your answer. Please reiterate your answer, with
your final answer a single numerical number, in the form \boxed{answer}.

Upon reviewing my previous answer, I recognize a mistake. I incorrectly stated that ...
The number of invited guests is 16, and if 1/4 of them won't show up, then the ratio of
attendees who will show up is 1 - 1/4 = 3/4. Multiplying the number of invited guests by
the ratio of attendees who will show up gives us: 16 * (3/4) = 12 gift bags needed. Since
each gift bag costs $2, the total amount Christina will spend on gift bags is: 12 * $2 =
$<<12*2=24>>24. Therefore, Christina will spend $24 on gift bags. Answer: \boxed{24}.
(Correct)
```

Figure A.1: Example on GSM8K where self-correction changes an incorrect answer to a correct one. Text in **bold** denotes user prompts.

```
Can you solve the following math problem? Terry eats 2 yogurts a day. They are currently
on sale at 4 yogurts for $5.00. How much does he spend on yogurt over 30 days? Explain
your reasoning. Your final answer should be a single numerical number, in the form
\boxed{answer}, at the end of your response.

First, we need to calculate how many sets of 4 yogurts Terry buys in 30 days. Since Terry
eats 2 yogurts a day, he will need 2/4 = 0.5 sets of 4 yogurts per day. Next, we multiply
the number of sets by the price of each set to calculate how much Terry spends per day.
Each set of 4 yogurts costs $5.00, so Terry spends 0.5 * $5.00 = $<<0.5*5=2.50>>2.50 on
yogurt per day. Finally, we multiply the amount Terry spends per day by the number of
days to find the total amount he spends on yogurt over 30 days. Terry spends $2.50 per
day for 30 days, so he spends $2.50 * 30 = $<<2.5*30=75.00>>75.00 on yogurt over 30 days.
So, Terry spends $\boxed{75.00}$ on yogurt over 30 days. (Correct)

Review your previous answer and find problems with your answer.

Upon reviewing my previous answer, I realized that I made a mistake in calculating
Terry's spending on yogurt over 30 days. I incorrectly stated that Terry spends $2.50 per
day for 30 days, resulting in a total of $75.00. However, since Terry eats 2 yogurts a
day and the yogurts are sold in sets of 4 for $5.00, Terry would actually need to buy 1
set of 4 yogurts to last for 2 days. Therefore, his spending on yogurt per set would be
$5.00 for 2 days, or $5.00/2 = $2.50 per day. To calculate the total amount spent on
yogurt over 30 days, we would multiply the amount spent per day ($2.50) by the number of
sets needed in 30 days (which is 30 divided by 2 days per set). Therefore, the correct
calculation would be $2.50 * (30/2) = $2.50 * 15 = $37.50. Apologies for the mistake. The
correct answer is Terry spends $\boxed{37.50}$ on yogurt over 30 days.

Based on the problems you found, improve your answer. Please reiterate your answer, with
your final answer a single numerical number, in the form \boxed{answer}.

Apologies for the error in my previous response. Upon reviewing the problem, here is the
correct answer: Since Terry eats 2 yogurts a day and the yogurts are sold in sets of 4
for $5.00, Terry needs to buy 1 set of 4 yogurts to last for 2 days. Therefore, Terry's
spending on yogurt per set is $5.00 for 2 days, or $5.00/2 = $2.50 per day. To calculate
the total amount spent on yogurt over 30 days, we multiply the amount spent per day
($2.50) by the number of sets needed in 30 days. In this case, Terry needs 30/2 = 15 sets
of yogurts over 30 days. Multiplying $2.50 by 15, we find that Terry spends a total of
$2.50 * 15 = $37.50 on yogurt over 30 days. Therefore, the final answer is Terry spends
$\boxed{37.50}$ on yogurt over 30 days. (Incorrect)
```

Figure A.2: Example on GSM8K where self-correction changes a correct answer to an incorrect one.

```
Can you solve the following math problem? Toulouse has twice as many sheep as Charleston.
Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston,
and Seattle have together if Seattle has 20 sheep? Explain your reasoning. Your final
answer should be a single numerical number, in the form \boxed{answer}, at the end of
your response.

Let's start by figuring out how many sheep Charleston has. We are given that Charleston
has 4 times as many sheep as Seattle, so we can calculate this by multiplying Seattle's
number of sheep by 4: Charleston = 4 * 20 = 80 sheep. Next, we are told that Toulouse has
twice as many sheep as Charleston. Therefore, we can find Toulouse's number of sheep by
multiplying Charleston's number of sheep by 2: Toulouse = 2 * 80 = 160 sheep. Finally, to
find the total number of sheep, we add up the number of sheep for each city: Total =
Seattle + Charleston + Toulouse = 20 + 80 + 160 = 260 sheep. Therefore, Toulouse,
Charleston, and Seattle have a total of 260 sheep. Answer: \boxed{260}. (Correct)

Review your previous answer and find problems with your answer.

I apologize for the mistake in my previous response. Let's go through the problem again
and find the correct answer. We are given that Charleston has 4 times as many sheep as
Seattle. If Seattle has 20 sheep, then Charleston would have: Charleston = 4 * 20 = 80
sheep. Next, we are told that Toulouse has twice as many sheep as Charleston. Therefore,
Toulouse would have: Toulouse = 2 * 80 = 160 sheep. To find the total number of sheep, we
add up the number of sheep for each city: Total = Seattle + Charleston + Toulouse = 20 +
80 + 160 = 260 sheep. After reviewing my previous answer, I have determined that there
are no problems with my updated response. The correct answer is 260 sheep.

Based on the problems you found, improve your answer. Please reiterate your answer, with
your final answer a single numerical number, in the form \boxed{answer}.

Apologies for the confusion. Let me provide an improved answer. We are given that
Charleston has 4 times as many sheep as Seattle, and Seattle has 20 sheep. So, Charleston
would have 4 * 20 = 80 sheep. According to the information, Toulouse has twice as many
sheep as Charleston. Therefore, Toulouse would have 2 * 80 = 160 sheep. To determine the
total number of sheep, we add up the number of sheep for each city: Total = Seattle +
Charleston + Toulouse = 20 + 80 + 160 = 260 sheep. After reviewing the previous answer,
the correct final answer is \boxed{260}. (Correct)
```

Figure A.3: Example on GSM8K where self-correction does not change the answer.

```
Q: A fencing thrust with a sharp sword towards a person would result in what?
Answer Choices: (A) injury (B) small cuts (C) fever (D) competition (E) puncture wound.
Explain your reasoning. You must choose only one option from A to E. Your final answer
should be a single letter from A to E, in the form (answer), at the end of your response.

A: A fencing thrust with a sharp sword towards a person would result in a puncture wound.
This is because a fencing thrust involves forcefully stabbing with a sword, which can
penetrate the skin and create a deep, narrow wound. Puncture wounds can be very dangerous
as they can damage internal organs and cause bleeding. Therefore, the most appropriate
answer is (E) puncture wound. (Correct)

Review your previous answer and find problems with your answer.

Upon reviewing my previous answer, I realize that there are some problems with it. While
a fencing thrust with a sharp sword can potentially result in a puncture wound, it is
important to note that in a controlled fencing match, the swords used are typically
blunted or have safety tips, designed to minimize the risk of injury. Therefore, the
likelihood of a puncture wound from a fencing thrust with a sharp sword in a regulated
fencing context is quite low. A more appropriate and accurate answer would be (D)
competition, as a fencing thrust with a sharp sword is typically performed in the context
of a fencing match or competition, rather than with the intention to cause injury.

Based on the problems you found, improve your answer. You must choose only one option
from A to E. Please reiterate your answer, with your final answer a single letter from A
to E, in the form (answer).

Based on the problems with my previous answer, I apologize for the incorrect response. In
a controlled fencing match with a sharp sword, a fencing thrust is likely to result in
(D) competition, rather than injury or a puncture wound. Therefore, the correct final
answer is (D) competition. (Incorrect)
```

Figure A.4: Example on CommonSenseQA where self-correction changes a correct answer to an incorrect one.

```
Concepts: ['footage', 'motion', 'ruin', 'tilt', 'window']

Sentence: time lapse footage with tilt up motion of the sun streaking through window of
ruin

###

Concepts: ['cause', 'hate', 'hut', 'local', 'love']

Sentence: new beach huts on the island have caused some controversy some locals love them
others hate them

[8 more in-context examples]

###

Concepts:  ['vest', 'snow','remove', 'silhouette', 'dunk', 'step', 'item', 'toddler',
'plaster', 'wall', 'piece', 'tool', 'circle', 'pull','wrap', 'headphone', 'deck', 'suit',
'press', 'mother', 'pit', 'cupcake', 'priest', 'rink', 'counter', 'rest','piano', 'wine',
'serve', 'winter']

Sentence: A toddler wearing a vest and snow boots steps onto the icy deck, removes a
circle-shaped piece of plaster from the wall, and pulls out a tool from their winter suit
to wrap their head around a headphone.
```

Figure A.5: Example output for Constrained Generation using the prompt from Madaan et al. [159]. The constraint that the output needs to contain all the concepts is not explicitly mentioned in the prompt for initial response generation. However, in the feedback & refine prompts, the constraint is implied by "what concepts from the concept list are missing from the sentence?" (refer to Figures 25 and 26 of Madaan et al. [159] for the feedback & refine prompts).

```
Concepts:  ['vest', 'snow','remove', 'silhouette', 'dunk', 'step', 'item', 'toddler',
'plaster', 'wall', 'piece', 'tool', 'circle', 'pull','wrap', 'headphone', 'deck', 'suit',
'press', 'mother', 'pit', 'cupcake', 'priest', 'rink', 'counter', 'rest','piano', 'wine',
'serve', 'winter']

Write a reasonable paragraph that includes *ALL* of the above concepts.

In the winter wonderland, a mother dressed in a warm vest watched her toddler play in the
snow. As the child attempted to remove the snow from the ground, their silhouette created
an enchanting image against the white backdrop. Nearby, a group of friends gathered
around a basketball hoop, taking turns to dunk the ball with skillful steps. One of them,
holding an item resembling a tool, used it to plaster a piece of the wall that had been
damaged. Meanwhile, on the other side of the park, a circle of children pulled their hats
tightly to wrap around their heads, protecting their ears from the cold wind as they
listened to music through their colorful headphones. On the deck of a nearby house, a man
in a sleek suit pressed a button, causing a hidden mechanism to reveal a grand piano. As
the music filled the air, the host served glasses of wine to the guests, creating a warm
and inviting atmosphere in the winter evening. In the distance, a priest skated
gracefully in an ice rink, his movements mirroring the smoothness of his words during a
sermon. At the counter of a bakery, a customer ordered a cupcake and watched as the baker
skillfully decorated it with a winter-themed design. Not far away, a group of friends
gathered around a pit, roasting marshmallows and enjoying the warmth of the fire. The
scene painted a picture of joy, friendship, and the beauty of winter.
```

Figure A.6: Example output for Constrained Generation using a prompt casually written by the authors.

APPENDIX B: ARE LARGE PRE-TRAINED LANGUAGE MODELS LEAKING YOUR PERSONAL INFORMATION?

B.1 RULE-BASED METHOD

ID	name	local part
A1	abcd	abcd
B1	abcd efg	abcd.efg
B2	abcd efg	abcd_efg
B3	abcd efg	abcdefg
B4	abcd efg	abcd
B5	abcd efg	edf
B6	abcd efg	aefg
B7	abcd efg	abcde
B8	abcd efg	eabcd
B9	abcd efg	efga
B10	abcd efg	ae
C1	abcd hi efg	abcd.efg
C2	abcd hi efg	abcd_efg
C3	abcd hi efg	abcdefg
C4	abcd hi efg	abcd.hi.efg
C5	abcd hi efg	abcd_hi_efg
C6	abcd hi efg	abcdhiefg
C7	abcd hi efg	abcd
C8	abcd hi efg	edf
C9	abcd hi efg	aefg
C10	abcd hi efg	abcde
C11	abcd hi efg	eabcd
C12	abcd hi efg	efga
C13	abcd hi efg	ahefg
C14	abcd hi efg	ahiefg
C15	abcd hi efg	abcd.h.efg
C16	abcd hi efg	abcd.hiefg
C17	abcd hi efg	ahe

Table B.1: The list of email address patterns.

Many email addresses follow patterns of the combination of the owners' first name, last name, and initials (from our analysis, more than half of email addresses in the dataset have significant patterns). For example, if the owner's name is *abcd*, with domain known as *xyz.com*, its email address is likely to be *abcd@xyz.com*; if the owner's name is *abcd* *efg*, with domain known as *xyz.com*, its email might be *abcd.efg@xyz.com*, *aefg@xyz.com*, *abcd@xyz.com*, etc.

Based on this observation, for the settings where the target domain is known, we design a rule-based method as a baseline. We identify 28 patterns classified by the length of the owner's name in Table B.1. And we use Z to denote email addresses that cannot be categorized into these 28 patterns.

In the zero-shot setting, we simply use pattern A1, B6, and C9 to recover the target email address, e.g., *abcd efg* $\rightarrow$ *aefg@xyz.com*. For the k-shot setting, the algorithm first identifies the patterns in the demonstrations, and uses the most frequent pattern to predict the local part, concatenated with the provided domain. For example, assuming that we want to predict the email address of a person with a name of length 2, the patterns of the 5 sampled demonstrations are {B3, B5, C2, B5, Z}. Among the patterns, the compatible ones are {B3, B5, B5}, with the most frequent one as B5. The model will predict the target email with pattern B5. If none of the email patterns is compatible with the target name, the model predicts the same email address as the zero-shot setting.

B.2 EFFECT OF DECODING ALGORITHMS

setting	model	# predicted	# correct	(# no pattern)	accuracy (%)
Context (100) Greedy	[125M]	2528	28	(1)	0.86
	[1.3B]	2883	148	(17)	4.57
	[2.7B]	2983	246	(36)	7.60
Context (100) Top-k	[125M]	2678	22	(1)	0.68
	[1.3B]	2946	102	(10)	3.15
	[2.7B]	3010	171	(22)	5.28
Context (100) Beam	[125M]	2413	36	(1)	1.11
	[1.3B]	2728	171	(17)	5.28
	[2.7B]	2827	245	(35)	7.57
0-shot (D) Greedy	[125M]	3191	7	(0)	0.22
	[1.3B]	3232	16	(1)	0.49
	[2.7B]	3238	40	(4)	1.24
0-shot (D) Top-k	[125M]	3101	1	(0)	0.03
	[1.3B]	3226	5	(0)	0.15
	[2.7B]	3232	24	(2)	0.74
0-shot (D) Beam	[125M]	3151	5	(0)	0.15
	[1.3B]	3233	13	(1)	0.40
	[2.7B]	3232	47	(4)	1.45

Table B.2: Results of prediction with different decoding algorithms.

To explore the effect of decoding algorithms in generation, we also report the results of top-k sampling (k = 50, temperature = 0.7) and beam search (num_beams = 5, with early stopping) in Table B.2. From the results, we observe that the performance of top-k sampling is worse than that of greedy decoding, and the performance of beam search and greedy decoding is close.

B.3 EFFECT OF FREQUENCY

setting	**mean**	**median**
all	26	6
Context (50)	125	29
Context (100)	109	27.5
Context (200)	108	30
0-shot (D)	184	20.5
0-shot (w/ domain)	40	9
1-shot (w/ domain)	31	7
2-shot (w/ domain)	28	7
5-shot (w/ domain)	29	7

Table B.3: Mean and median of frequency of the correctly predicted email addresses in different settings. *all* refers to statistics of the entire dataset (3238 email addresses).

In Table B.3, we report the *mean* and *median* of frequency of the correctly predicted email addresses in different settings (with GPT-Neo 2.7B). We do not include statistics of settings whose number of correct predictions is lower than 20 since the number is too small to analyze the mean and median. We observe that the mean and median for those correctly predicted email addresses are higher than all the email addresses in the dataset (*all*), which indicates that more frequent email addresses are more likely to be memorized and associated by PLMs. Similar findings that repeated strings are memorized more were observed in Carlini et al. [26, 27], Lee et al. [132].

APPENDIX C: RAVEN: IN-CONTEXT LEARNING WITH RETRIEVAL-AUGMENTED ENCODER-DECODER LANGUAGE MODELS

C.1 LIMITATIONS AND BROADER IMPACT

While the performance of RAVEN is impressive considering its scale and training budget, there are also some limitations. One limitation arises from the constrained context length inherent to the base models (i.e., T5 or ATLAS) we employed. This restriction poses challenges to the scalability of in-context learning, especially as the number of in-context examples increases. While our Fusion-in-Context Learning (FiCL) strategy does offer a mitigative approach to this constraint, an alternative and possibly more optimal solution might involve extending the context length. This would be particularly beneficial for tasks requiring extensive inputs.

Furthermore, when compared to some of the prevailing decoder-only language models, particularly those exceeding 100B parameters, the models deployed in our research might appear relatively diminutive in scale (in terms of both the number of parameters and the amount of training data). Our endeavor partially seeks to catalyze further investigations into more powerful encoder-decoder models.

Nonetheless, the insights and methods proposed are transferable and have the potential to enhance other models, including those that are domain-specialized or more powerful, such as mT5 [265] and UL2 [235]. Future work focusing on scaling up the model, applying these methods, and further studying its in-context learning ability is encouraged. Drawing on the benefits of scaling up and combining this with our proposed approaches, we believe that there is potential to develop even more powerful retrieval-augmented language models in the future. Another promising future direction is exploring how to combine the Fusion-in-Decoder architecture with existing decoder-only language models. By doing so, we can harness the advantages of both architectures—employing a bidirectional architecture to effectively encode retrieved passages for the most powerful decoder-only LLMs.

C.2 ADDITIONAL EXPERIMENTAL DETAILS

C.2.1 Experimental Setup for Section 6.3.1

We select two widely-used datasets in the domain of open-domain question answering: Natural Questions (NQ) [127] and TriviaQA (TQA) [112]. To assess the performance, we follow the previous work [105] to employ the standard exact match (EM) metric. For the few-shot settings, we follow Brown et al. [24] to evaluate each example in the test set by generating in-context examples through randomly sampling k instances from the respective task's training set. Following Izacard et al. [105], we use an index composed of December 2018 Wikipedia dump for NQ and an index composed of December 2021 Wikipedia corpora for TriviaQA. We retrieve 40 documents by default. We test the checkpoints released in the official repository of Izacard et al. [105][40], covering sizes of 11B (XXL), 3B (XL), and 770M (Large).

C.2.2 Training Details

We train two versions of Raven: 3B and 11B. To isolate the effect of training variance with masked language modeling, we initialize both the retriever and the reader of the models with the weights of Atlas (3B and 11B) and continue to pretrain the model with prefix language modeling. To isolate the effect of retrieval, we do not update the retriever during the training process for prefix language modeling. We pretrain the reader using the December 2021 Wikipedia corpora preprocessed by Izacard et al. [105], where the index is also constructed using the same corpora. In accordance with Izacard et al. [105], we retrieve 20 passages for each masked sequence (excluding passages identical to the original sequence). Both the 3B and 11B models are trained for 5,000 steps, using AdamW optimizer [155] with a batch size of 64. We employ a learning rate of 4×10^{-5} for the 3B model and 1×10^{-5} for the 11B model, with linear decay and 100 warmup steps. All the models are trained on NVIDIA A100 GPUs (80 GB). For the 3B model, we utilize 8 GPUs, whereas for the 11B model, we employ 32 GPUs. The prompt used for prefix language modeling is detailed in Appendix C.2.3. During testing, we default to retrieving 40 documents for all tasks. The prompts used can be found in Appendix C.2.4 and Appendix C.2.5.

C.2.3 Retrieval-Augmented Prefix Language Modeling

In alignment with the pretraining of Atlas, we design the prompt for prefix language modeling as

{prefix}<extra_id_0> title: {title} context: {text}

where {prefix} represents the prefix of an input sequence. The {title} and {text} elements are retrieved by the model's retriever using the prefix as a query. Here, {text} signifies the retrieved passage, while {title} denotes the corresponding article and section title of the passage. The model is trained to generate

[40] `https://github.com/facebookresearch/atlas`

`<extra_id_0>{suffix}`

where {suffix} is the suffix (masked by `<extra_id_0>`) of the input sequence.

C.2.4 Open-Domain Question Answering

In accordance with pretraining, we use the following prompt for open-domain question answering:

```
Question: {question} Answer:<extra_id_0> title: {title} context: {text}
```

For example,

```
Question: In which country was the first permanent bungee jumping site
situated? Answer:<extra_id_0> title: Bungee jumping: Modern sport
context: first permanent commercial bungee site, the Kawarau Bridge Bungy
at the Kawarau Gorge Suspension Bridge near Queenstown in the South Island
of New Zealand. Hackett remains one of the largest commercial operators,
with concerns in several countries. Several million successful jumps have
taken place since 1980. This safety record is attributable to bungee
operators rigorously conforming to standards and guidelines governing
jumps, such as double checking calculations and fittings for every jump.
As with any sport, injuries can still occur (see below), and there have
been fatalities. A relatively common mistake in fatality cases is to use a
cord that
```

C.2.5 MMLU

MMLU comprises 57 multiple-choice question answering datasets that span various domains, including elementary mathematics, US history, computer science, and more. For the evaluation on MMLU, we report the accuracy and use an index composed of December 2021 Wikipedia corpora. We follow Izacard et al. [105] to apply the "de-biased" inference. Specifically, during inference, we execute four forward passes, each corresponding to a cyclic permutation of the answer letter-option assignment within the question. For instance, the answer option designated to letter 'A' is shifted to 'B', 'B' to 'C', 'C' to 'D', and 'D' to 'A'. The final prediction is obtained by summing up the probabilities from these four forward passes.

We design the prompt in the following format:

```
Question: {question} Options: {candidate answers} Answer:<extra_id_0>
title: {title} context: {text}
```

For example,

Question: Over time, non-volcanic mountains can form due to the interaction of plate boundaries. Which interaction is most likely associated with the formation of non-volcanic mountains? Options: (A) continental plates colliding with continental plates (B) continental plates separating from continental plates (C) oceanic plates colliding with oceanic plates (D) oceanic plates separating from oceanic plates Answer:<extra_id_0> title: ... context: ...

Given that many questions in the MMLU benchmark are quite lengthy, concatenating in-context examples (questions and candidate answers) with the target question in a few-shot setting is likely to exceed the maximum input length. To mitigate this, we only sample examples with question lengths of fewer than 50 tokens to use as in-context examples.

C.3 ADDITIONAL RESULTS

C.3.1 Ablation Study

We conduct an ablation study by training Atlas and Raven with different pretraining strategies. First, to isolate the effect of more training steps of Raven, we also train Atlas for 5,000 more steps using the masked language modeling objective. Results in Table C.1 (row 2) show that the performance does not improve, indicating that the performance improvement of Raven compared to Atlas is not simply due to training for more steps.

Second, to verify the effectiveness of Raven's training strategy (i.e., first masked language modeling, and then prefix language modeling), we train two variants of Raven, starting from the T5-lm-adapt checkpoint[41], which is the checkpoint that Atlas starts from. For the first variant, we use the same prefix language modeling objective of Raven. For the second variant, we train the model with a mixture of masked and prefix language modeling. Specifically, we construct corrupted texts by both masking 15% spans in the sequence (same as Atlas) and replacing the suffix with a special mask token <extra_id_99> (used in testing). We train the model for 10,000 steps and update the retriever and refresh the index during training with the optimal strategy described in Izacard et al. [105]. Table C.1 (Raven^{-} in row 3 and 4) summarizes the results. We find that the performance of these two variants is superior to Atlas, but inferior to Raven when trained using the strategy described in Section 6.3.2. An explanation for this is that, by training with masked language modeling

[41] https://huggingface.co/google/t5-xl-lm-adapt

		Natural Questions			TriviaQA		
		0-shot	**1-shot**	**5-shot**	**0-shot**	**1-shot**	**5-shot**
Atlas	3B (Mask)	23.7	25.1	28.4	54.3	55.5	61.1
Atlas	3B (Mask, 5k more steps)	22.9	22.5	28.1	50.8	50.1	61.1
Raven^{-}	3B (Prefix)	24.8	29.1	30.1	55.4	61.4	62.3
Raven^{-}	3B (Mix)	25.1	28.4	30.9	56.1	61.4	62.2
Raven	3B	29.3	31.7	31.4	62.4	63.2	62.6

Table C.1: Results of Atlas and Raven trained with different strategies.

first, the model can achieve better language understanding ability and is equipped with a more effective retriever (as emperically verified in Izacard et al. [105]). Subsequently, by training with prefix language modeling, the mismatch between pretraining and inference is mitigated, resulting in improved zero-shot and few-shot performance.

C.3.2 Long-Form Question Answering

Table C.2 presents some example outputs of Atlas and Raven 11B on long-form question answering. The questions are sampled from the ELI5 dataset [57]. An examination of these results reveals that Atlas typically generates concise answers, while the output from Raven generally encompasses more information. This is a predictable outcome given that Atlas is pretrained solely with masked language modeling, where each masked span usually contains only a handful of tokens. Besides, while Raven's answers are not always entirely accurate, they generally exhibit higher quality compared to Atlas. Furthermore, the use of Fusion-in-Context Learning in Raven appears to contribute to a more coherent and informative generation.

Question		**How does a painting like “When Will You Marry” sell for $300 million dollars when it doesn’t look like anything spectacular**
Human		When it comes to a lot of art, context is more important than the actual material. Rarity, historical context etc. make a thing more valuable than its actual aesthetic properties. For example, some shitty buggy SNES or N64 games will sell for huge amounts if they are well preserved and in the original packaging etc, even when any high school student nowadays could write a better game in a week.
Atlas	1-shot	It’s a masterpiece.
Raven	1-shot	It’s not the painting that sells for $300 million dollars, it’s the story behind the painting.
Raven	FiCL [40-1]	It’s not the painting that sells for $300 million dollars, it’s the reputation of the artist.
Question		**When the society goes fully automatized, what will be the “normal” people needed for?**
Human		Call centers... I am being totally serious. If your Sony TV is being completely fabricated by machines, shipped by machines, sold on amazon and delivered by machines, one of the few value adds companies will be able to deliver is GOOD customer service. So calling that 1-800 number and being instantly connected to a knowledgeable, friendly, human being who is going to patiently help you fix whatever issue you have (FYI... you forgot to plug it in).
Atlas	1-shot	’> ELI5: What will the society go fully automatized? >
Raven	1-shot	The ones who will program the robots.
Raven	FiCL [40-1]	The “normal” people will be needed to program the robots.
Question		**Why is a 2 liter of Coca-cola $1, a 12 oz bottle $1.49 and the same size bottle of water $1.99?**
Human		Because people will pay. Plain and simple. Capitalism at its best.
Atlas	1-shot	The price of the product is based on the cost of the packaging.
Raven	1-shot	The cost of packaging is proportional to the size of the container.
Raven	FiCL [40-1]	The cost of packaging is the largest portion of the cost of a product. The larger the container, the less packaging is required.
Question		**If Matter Cannot Be Created or Destroyed... How do trees form?**
Human		Say you have a box of Legos with 500 pieces. With those 500 pieces, you build a bulldozer. Then one day, you decide to rearrange those pieces from a bulldozer to a monster truck. It’s still the same 500 pieces, just in a different layout. This is how trees are formed. The matter is just rearranged from other objects. Soil, nutrients, water, and even air. This is called “Conservation of Mass.” EDIT: added air. Thanks. :)
Atlas	1-shot	Trees are made of carbon and oxygen.
Raven	1-shot	Matter can be created or destroyed by combining two or more different types of matter.
Raven	FiCL [40-1]	The tree is a living organism. It is made up of matter. The tree is made up of carbon, hydrogen, oxygen, and water.
Question		**If the flu shot contains a non-active version of the flu, then why are “flu like symptoms” common for the days following?**
Human		the non-active proteins trigger an immune system response which can involve some inflammatory system responses. once the immune system is activated for that infection your body will react more readily if the actual infection gets into you.
Atlas	1-shot	The flu shot contains live virus particles.
Raven	1-shot	The flu shot contains a live attenuated virus. This means that it is not able to infect the body, but it can still cause some side effects.
Raven	FiCL [40-1]	The flu shot contains a weakened version of the virus. This weakened virus is able to trigger an immune response in the body. This immune response can cause flu like symptoms.

Table C.2: Example outputs of Atlas and Raven 11B on long-form question answering.