Reasoning in the Time of Scaling

Laura Ruis

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Department of Computer Science
University College London

January 28, 2026

To my parents, who gave me a childhood so secure I learned to choose curiosity over safety.

Acknowledgements

I spent most of my life more interested in other things than science, even taking a few years after high school to work in restaurants across Europe before finally going to university. All that changed when I encountered Artificial Intelligence at the end of my undergraduate degree at the University of Amsterdam. I switched to AI for my MSc, marking the beginning of my passion for the topic. I feel incredibly lucky to have had many people shape my career.

I am deeply indebted to those who believed in me early on, when I had no programming skills to speak of, no research experience, and only a shallow understanding of AI. The first was Jasmijn Bastings, who took me on as an MSc student at the UvA Institute for Logic, Language and Computation, where she taught me the basics of experimental design, reproducing existing work, and testing hypotheses without succumbing to graduate student descent on the test set. Those lessons still serve me every day. Then Brenden Lake took me on as an intern at Facebook AI Research for my first full research project. From him I learned to think about what was still missing from AI: the richness of human cognition. He introduced me to cognitive science, and the time at FAIR and NYU was formative. The questions I encountered there, about what it means to truly understand, to learn, to generalise, run through every chapter of this thesis.

These experiences landed me a PhD position with Edward Grefenstette and Tim Rocktäschel at UCL DARK, to whom I owe the most. All four years of my PhD were a pleasure, and I owe that to the lab Ed and Tim built. Beyond a place of technical excellence, filled with people with a passion for AI, they created a safe place and looked out for students' wellbeing. They also taught me to think deeply about the potential impact of my research, placing it within the broader trajectory of the field, and motivated us to do research that would remain relevant as AI systems grew more powerful. Beyond that, each shaped me in distinct ways. Tim, I feel lucky to have absorbed your unique approach to AI research: the north-star method, focusing on long-term goals and working toward them step by step. This approach has cured me of worrying about being scooped, and it is something I hope to carry forward if I ever get to build my own lab. I also owe you my external brain in Obsidian, without which I feel like only half a researcher. Ed, your incredibly broad range of knowledge, and way of asking and answering research questions, has shaped the core direction of many chapters in this thesis. Also, making me read Philosophical Investigations in my first year shaped my thinking in lasting ways.

The UCL DARK lab is full of remarkable people, and I am grateful to all of them for being cracked researchers to look up to: Akbir, Robert, Mika, Zhengyao, Minqi, Yingchen, Davide, Nathan, Ahmet, and Ross. My desk was right next to that of Akbir Khan, who became my best friend in London. Anyone who knows Akbir knows he is the life energy of every group, and I love you for making my PhD and time in London fun. I am also grateful to Rob Kirk, who became a dear friend and slowly convinced me of the importance of AI safety. I feel safer working on AI knowing that you dedicate your career to it.

A non-trivial part of my PhD was spent interning at Cohere, and I owe an important part of this thesis to my manager Max Bartolo. Max and I pursued an overly ambitious project for an internship, and it became one of my favourite collaborations. Working with an adviser who was involved in every aspect — from in-the-weeds XLA bugs to high-level research questions — was an absolute pleasure. Without your constant belief in a project that lasted much longer than we had anticipated, I would have never been able to answer the research question in Chapter 3 of this thesis. As a side effect, my time at Cohere significantly enriched my life in London, where I met many amazing people, some of whom became collaborators: Max Mozes, Sid, Dwarak, and Acyr. In Acyr I found more than a collaborator. Getting to know the hidden range of your passions, your rare combination of leading teams with humour, real warmth, and deep technical knowledge, has been a treat I did not see coming, but one I hope to keep learning from.

I owe thanks to many other collaborators and friends in AI I have met along the way. Ekdeep, you may know the most about how AI works of anyone I know, and I am happy to call you my friend. Jonny, working with you was the first time another PhD student wanted to build on my work in that way, and led to a project driven by you that I am very proud of and covers a whole chapter in this thesis. I want to thank you for that. Others who have touched my PhD time in important ways are Ethan, Emile, Evgenii, Silvia, Stella, Juhan, Arduin, Herbie, Saeed, Robert de Vries and Mark Hill, Sara Hooker, Marco Baroni, Julia Proskurnia, Jyrki Alakuijala, Martin Bruse, and Jacob Andreas. Moreover, I am grateful to the students that were willing to work with me and teach me about advising research: J, Itamar, Yi, Alex, Can, Russell, Efstathios, and Rob. Working with you has given me some of the highlights of my career.

On the personal side, the most important presence during my PhD has been Philipp, whose love and support carried me through my time as a researcher. Together we dealt with the stress of our PhDs by climbing mountains without preparation or the necessary gear (I'll never forget the patience with which you countered my mental breakdowns on Maja Jezerce). From you I learned to deeply question my assumptions, never satisfied with a particular level of understanding. You made me a better researcher and thinker. Only when you moved to London did it truly start to feel like home.

Another person who made London feel more like home is Lena. When you tried to headhunt me for a startup and realised that was not going to happen, we became friends instead. Our endless conversations over dinner (cooked by you) became the highlight of my time here. Your combination of warmth, intelligence, curiosity, and raw ambition is rare, and I feel lucky to have you as a friend.

I am deeply grateful to my dear old friends who stood by me while I pursued my PhD. Words cannot express how much it means to me that we have stayed friends while I moved away, always coming to visit me in the different countries I moved to, always being my biggest fans: Esther, Inge, Ary, Emu, Robin, Aweng, and Frederieke. Besides the cherished times that I got to see all of you in person, I never knew how important group chats could be for my general well-being (one of them unusually enthusiastic about edge cases). Some of you have since become parents to my favourite additions to humanity: Joes, Lou, Phoebe, and Robin. Watching your lives reshape around them has been one of the joys of these years.

Last but not least, I want to thank my family, my unwavering foundation: Elise, Henk, Ruud, Frank, Bella, Hasnaa, Emma, Kasper, and Sander. The ability to come home to all of you makes my life abroad feel grounded and safe. Bella, my sister, I feel attached to you through an invisible string, while we both live life differently. You uncover every stone of life and show me what's beneath it. Because of you I have not only progressed professionally these past four years, but am also learning to be a fuller human being. Frank, my brother, you are the epitome of an investigator, turning everything your mind touches into a deep rabbit hole. I admire your dedication to the full truth, and the way you're applying that to your personal journey. Mom, you taught me to be non-judgemental and kind. You showed me what it means to dedicate your work to others, taking on cases no one else would fight for. Dad, I owe you my mind. I greatly admire you in your profession and as a writer, and the humour and competence you bring to both. I owe both of you everything I get to do with my life.

Declaration

I, Laura Eline Ruis, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Laura Eline Ruis

Abstract

This thesis investigates the nature of reasoning in large language models (LLMs) within the context of the compute scaling paradigm. I define reasoning as the process of deriving novel information from existing knowledge, an ability that is difficult to evaluate when models train on trillions of tokens of data.

Across four studies, I progressively develop a characterisation of LLM reasoning. First, I empirically demarcate LLM reasoning from memorisation by showing that models learn distinct strategies from pre-training data for mathematical and factual retrieval tasks. Reasoning involves generalising from shared patterns across questions, while factual retrieval relies on specific instances. The second study situates LLM reasoning further yet from memorised patterns. When LLMs are fine-tuned to auto-regressively predict source code tokens for previously unseen programs, they also develop the ability to evaluate these programs for inputs. This emergence demonstrates that models extract computational principles during training that abstract away from specific inputs and generalise beyond their original context. Turning my attention away from reasoning that follows strict axioms, in the third study I show LLMs also acquire social reasoning abilities during large-scale training. Models demonstrate human-level performance in inferring implicit communicative intentions from ambiguous text, suggesting pragmatic understanding. This ability emerges during large-scale next-token prediction, strengthens with model scale, and is most improved during post-training. However, my final study reveals that the socio-cognitive mechanisms that underpin such human pragmatic understanding may not have conclusively emerged. This underscores how machine reasoning may differ from human reasoning. Taken together, these findings characterise LLM reasoning as a versatile computational process that emerges with scale and generalises beyond training data to novel contexts, highlighting the broader potential of the compute scaling paradigm.

Impact Statement

The current economy is strongly oriented toward investment in artificial intelligence [NYT25]. In equity markets, the performance of the S&P 500 has been disproportionately driven by a small group of technology companies, often referred to as the "Magnificent Seven" (including Apple, Microsoft, Alphabet, Amazon, Meta, Tesla, and NVIDIA) [FT25]. This concentration reflects an underlying bet that continued advances in the compute scaling paradigm will ultimately yield artificial intelligence with high return on investment. The realisation of these returns hinges critically on whether large language models (LLMs) — the central manifestation of the scaling approach — are capable of reasoning, understood here as the ability to generate novel conclusions from existing knowledge.

This thesis takes important initial steps toward addressing this fundamental question by developing methods to characterise the nature and boundaries of LLM reasoning capabilities. While these findings contribute to understanding LLM capabilities, they represent early progress on a complex question that will require extensive further research to fully resolve. The emergence of reasoning abilities in constrained domains suggests promise for the scaling paradigm, but the distinct nature of machine reasoning processes and the limited scope of current investigations leave substantial uncertainty about whether and how these capabilities will translate to the broad, robust reasoning required for transformative economic impact.

Contents

List of Figures

List of Tables

List of Abbreviations

LLM . . . . . . .Large Language Model

AI . . . . .Artificial Intelligence

GPT . . . . . . .Generative Pre-Trained

SFT . . . . . . .Supervised Fine-Tuning

RL . . . . .Reinforcement Learning

RLHF . . . . . .Reinforcement Learning from Human Feedback

RLVR . . . . . .Reinforcement Learning from Verified Rewards

DPO . . . . . . .Direct Preference Optimisation

GRPO . . . . . .Group-Relative Policy Optimisation

PBB . . . . . . .Programming by Backpropagation

IO or I/O . . .Input/Output

ToM . . . . . . .Theory of Mind

TDA . . . . . . .Training Data Attribution

FOC . . . . . . .First-order Optimality Condition

LOO . . . . . . .Leave One Out

KFAC . . . . . .Kronecker-Factored Approximation to the Curvature

EKFAC . . . . .Eigenvalue-corrected Kronecker Factorization

PBO . . . . . . .Proximal Bregman Objective

PBRF . . . . . .Proximal Bregman Response Function

IHVP . . . . . . Inverse-Hessian-Vector product

FIM . . . . Fisher Information Matrix

GNH . . . . Gauss-Newton Hessian

i.i.d. . . . . Independent and Identically Distributed

Chapter 1

Introduction

> We alone among Earth's creatures inhabit a world of abstractions and impossibilities, brooding over what never was, musing over what might have been, and pondering what it will be like not to be. We tell stories about our real experiences and invent stories about imagined ones, and we use these stories to organize our lives. Living our life in this virtual world, we slowly realized that no other species can follow us here.
>
> *Terrence Deacon, The Symbolic Species (1997)*

For decades, language production and comprehension was considered a uniquely human behaviour [HCF02; PJ05; Fri17; TER19, inter alia.]. Today, large language models (LLMs) challenge this assumption. In the four years spanning my doctoral studies, LLMs evolved from experimental systems producing syntactically correct but often incoherent text into sophisticated models capable of generating contextually appropriate, semantically rich text. Most strikingly, they have become general-purpose tools integrated in millions of daily workflows[1], handling complex cognitive tasks that seemingly require reasoning and understanding [WHL23; WHL24; McL+24; Dub+24; Gem+25; Coh+25; Qwe+25, inter alia].

Much of these advancements are driven by scaling up computational resources during development, such as model and training data size [Kap+20; Hof+22]. If we define reasoning as the process of deriving new information from existing knowledge, the strong performance LLMs demonstrate on reasoning tasks alone is insufficient evidence of genuine reasoning capability. After all, we cannot

[1]ChatGPT had 800 million weekly active users in April 2025 according to `https://www.demandsage.com/chatgpt-statistics/`.

determine how closely these evaluation tasks resemble the training data. Do LLMs extract reasoning principles implicit in the data, developing capabilities that can generalise to novel problems, or does their apparent reasoning result from similar sequences encountered during training? The distinction is crucial: the former suggests a type of machine reasoning, while the latter implies a more limited process unlikely to contribute meaningfully to existing knowledge.

In this thesis, I aim to characterise LLM reasoning within the context of the compute scaling paradigm. I begin with a basic premise: the approach to reasoning must differ qualitatively from the memorisation of facts if it is to generalise beyond training data. Chapter 3 examines this distinction by contrasting how models learn mathematical reasoning with how they learn factual retrieval. In a large-scale experiment, my collaborators and me trace model predictions back to their origins in pre-training data, identifying which training examples contribute most strongly to specific responses. The analysis reveals a clear distinction: while factual responses draw on question-specific data, mathematical reasoning responses consistently rely on task-specific pre-training data, where the same underlying data influence multiple questions within a reasoning task. This pattern suggests models rely on what I term *procedural knowledge* for reasoning: knowledge patterns that transfer across multiple examples. For example, when solving for x in $5 = 3x + 2$ and $12 = 4x - 6$, models draw on similar data, indicating they acquire generalisable strategies rather than memorise individual solutions.

Having established that LLM reasoning differs from the memorisation of facts, in Chapter 4 I investigate the level of abstraction at which models acquire procedural knowledge. Can LLMs learn computational principles that abstract away from specific inputs? The previous chapter provides a clue: the most influential training sequences for reasoning contain not only worked examples with implicit procedural knowledge, but also *explicit procedures*: abstract, input-general representations of the task solution, such as code implementations or analytical formulas. An illustrative example of explicit procedural knowledge is the following program for solving linear equations:

```
def solve_linear(a, b, c):
    """Solve ax + b = c for x"""
    return (c - b) / a
```

However, the influence-based analysis in Chapter 3 identifies which training examples matter most without explaining why they are influential. In Chapter 4, we test whether LLMs can learn to apply procedures to novel inputs from abstract procedural representations alone. Specifically, when an LLM is fine-tuned by auto-regressively predicting the tokens of an unseen program's source code, does it also acquire the ability to evaluate the program for inputs, without ever seeing input-output pairs for that specific program? The results confirm models indeed to some extent learn to evaluate programs from generative training on their source code alone, provided they receive separate training on input-output pairs for programs implementing different procedures. Although their performance is far from perfect, they generalise uniformly across inputs from a single piece of code, and are able to evaluate compositions of programs encountered separately during training. The findings in this chapter confirm that LLMs extract abstract procedural knowledge from symbolic representations and apply these procedures to novel cases at inference time.

The previous two chapters characterise LLM reasoning as distinctly different from memorising patterns in pre-training data, showing how models extract generalisable strategies from worked examples and symbolic representations like code and mathematical formulas that can be applied in novel context. However, the analysis so far has focused exclusively on domains with clear rules and objective solutions: mathematics and logic follow strict axioms that make correct reasoning unambiguous. Human reasoning, however, extends beyond formal systems. Much of our cognitive work involves navigating the messy, context-dependent world of social interaction, where success depends on implicit knowledge of social norms, cultural conventions, and shared understanding. Social reasoning requires inferring unstated assumptions, interpreting communicative intent, and modelling the mental states of other agents; capabilities that seem to demand experiential learning rather than pattern extraction from text. The final chapters extend the characterisation of LLM reasoning to the social domain through two case studies.

I start by examining LLMs' ability to understand language pragmatics, the principle that utterances derive meaning not just from the literal words, but from contextual factors like conventions, shared beliefs, and background knowledge (Chapter 5). Consider Wittgenstein's famous example: the exclamation "water!' [Wit53]. Without context, it could signal a person lost in the desert finally seeing water in the distance, a homeowner's alarm at discovering a leak, or an environmental activist identifying humanity's greatest challenge. Can models interpret communicative intent, even when the relevant context is not explicitly given in the text? Testing a range of LLMs on tasks requiring interpretation of ambiguous language that humans readily understand, we find that certain models do possess this capability. The ability emerges in base models and strengthens with increased parameters, but the most significant improvement can be attributed to specific post-training methods designed to align LLMs with human values. Taken together, the findings suggest that the ability to infer contextual meaning can be learned from passive, mostly textual data.

Human understanding of implicit communicative intent is thought to depend on our ability to reason about other agents' mental states, a capacity known as theory of mind (ToM). In developmental psychology, children who successfully interpret the types of ambiguous language studied in Chapter 5 have typically reached key theory of mind milestones first. This raises a question: have LLMs similarly reached the milestones thought to precede pragmatic understanding? Chapter 6 presents a preliminary investigation into whether LLMs develop one of the earliest manifestations of theory of mind: infants' propensity to encode agent's goal-directed actions. In her seminal study, Woodward [Woo98] demonstrates that infants as young as 6–9 months encode the goal object of an agent's reaching event. When an agent repeatedly reaches for one object (a teddy bear) over another (a ball), infants exhibit surprise when the agent subsequently grasps the ball, suggesting they have formed expectations about the agent's object preferences. Crucially, when identical reaching motions are performed by an inanimate rod rather than a human hand, infants form no such expectations. This indicates they encode events differently based on agent animacy.

In Chapter 6, we investigate whether LLMs similarly demonstrate differential encoding of textual descriptions involving animate versus inanimate agents. We extend Woodward's original design by introducing an experimental control: scenarios where agents act accidentally rather than intentionally. This control condition tests whether models truly understand goal-directed behaviour or simply respond differently to animate versus inanimate actors regardless of intentionality. Our results prove inconclusive regarding a consistent machine theory of mind. While state-of-the-art models do show differential encoding for animate versus inanimate actors in some cases, they often respond similarly to both intentional and accidental actions by animate agents. This finding undermines the conclusion that models expect object preferences from animate agents. The study serves as a cautionary example for the field. In light of recent literature claiming that LLMs exhibit human-like theory of mind [Kos24], our findings highlight the importance of employing experimental controls when adapting human psychological tests for artificial intelligence.

The four studies in this thesis position LLMs as sophisticated systems in which machine reasoning has emerged across both mathematical and social domains. Their mathematical reasoning capabilities stem from extracting transferable procedures from sequential text and code data, enabling generalisation to novel problems and different levels of abstraction. Beyond formal domains with objectively correct solutions, LLMs acquire social reasoning abilities during large-scale training, allowing them to infer communicative intent in pragmatically ambiguous language. However, investigations into whether the foundational socio-cognitive mechanisms that underpin human pragmatic understanding have similarly emerged in LLMs yield inconclusive results. This underscores how machine reasoning may follow different pathways than human cognition while still achieving comparable performance on complex reasoning tasks.

The work presented here represents only initial steps towards uncovering the types of generalisations LLMs can achieve, and many questions remain unanswered. My findings demonstrate reasoning that generalises across different instances of mathematical tasks and from abstract symbolic representations to concrete applications. An obvious next question is: can models learn reasoning patterns that transfer across different tasks or fundamentally different forms of reasoning? For example, can the procedural knowledge we observe in formal, verifiable mathematical reasoning extend to the inductive reasoning that underpins empirical science, where evidence must be weighed, hypotheses formed, and conclusions drawn about questions with no directly verifiable answers?

If LLMs can indeed bridge the gap between deductive and inductive reasoning, then scaling up the current paradigm might enable far more ambitious applications than studied here. LLMs, having trained on much of the knowledge humans have produced, could potentially propose novel theorems, generate testable hypotheses, and contribute meaningfully to discovery in experimental domains like biology and physics, sciences where knowledge emerges through iterative experimentation rather than logical proof.

I started my doctoral studies in a pre-LLM era, where deep learning models failed to classify cows when these stepped onto a beach instead of their usual pastoral settings [BVP18], and more generally struggled to generalise out-of-distribution [LB18]. Within this context, it took me several years of research to accept that fundamentally different generalisation patterns emerge when models are trained at scale. Given the remarkable generalisations LLMs already make from simple self-supervised objectives, I remain cautiously optimistic that the current compute scaling paradigm can lead to systems capable of contributing genuinely new knowledge. After all, predicting the next token in an infinite stream of sequential data generated by the natural world can most efficiently be done by inferring the causal model underlying it.

1.1 Outline

The rest of this work is organised as follows. Chapter 2 provides comprehensive background on each of the concepts required to understand the contributions of this thesis, covering reasoning, LLM development, scaling laws, and influence functions. Chapter 3–6 present the core contributions introduced above. Chapter 7 briefly summarises these contributions, reflects on the wider impact of this work and the directions it opens for continued exploration beyond my doctoral studies.

1.2 List of Publications

The research presented in this thesis is based on the following publications:

- **Chapter 3 – Learning to Reason from Pre-training Data** [Rui+25]
  Ruis, L., Mozes, M., Bae, J., Kamalakara, S.R., Talupuru, D., Locatelli, A., Kirk, R., Rocktäschel, T., Grefenstette, E., Bartolo, M. (2025)
  ICLR 2025
  Procedural Knowledge in Pretraining Drives Reasoning in Large Language Models
  Personal contribution: project proposal and leadership, designing the experimental protocol, code, interpreting the results, paper writing.

- **Chapter 4 – Learning to Reason from Code Data** [Coo+26]
  Cook, J., Sapora, S., Ahmadian, A., Khan, A., Rocktäschel, T., Foerster, J., Ruis, L. (2025)
  ICLR 2026
  Programming by Backprop: One Instruction is Worth One Hundred Examples When Training LLMs
  Personal contribution: Main advisor. Jonny Cook and me developed the research question based on [Rui+25], Jonny drove implementations, together with Silvia Sapora. I was intimately involved in the overall direction, most of the experimental decisions, and paper writing.

- **Chapter 5 – Case Study: Pragmatic Reasoning** [Rui+23b]
  Ruis, L., Khan, A., Biderman, S., Hooker, S., Rocktäschel, T., Grefenstette, E. (2023)
  NeurIPS 2023 (spotlight)
  The Goldilocks of Pragmatic Understanding: Fine-Tuning Strategy Matters for Implicature Resolution by LLMs
  Personal contribution: project proposal and leadership, dataset development, code writing, human experiment.

- **Chapter 6 – Case Study: Theory of Mind** [Rui+23a]
  Ruis, L., Findeis, A., Bradley, H., Rahmani, H. A., Choe, K. W., Grefenstette, E., Rocktäschel, T. (2023)
  ToM workshop ICML 2023
  Do LLMs selectively encode the goal of an agent's reach?
  Personal contribution: project proposal and leadership, task development, interpretation of results, paper writing.

Chapter 2

Background

This chapter establishes the concepts required for understanding the thesis' contributions. I start with a brief note on reasoning, a sometimes controversial topic that humans have been thinking about since Aristotle. While I assume familiarity with the mathematical reasoning tasks discussed in Chapters 3 and 4 (arithmetic and basic control flow), I focus here on the social reasoning capabilities central to Chapters 5 and 6: pragmatic inference and theory of mind. I then discuss the basics of LLM development, covering pre-training, several post-training methods, and compute scaling. Finally, I provide background to understand influence functions, the method used in Chapter 3 to trace model responses back to their pre-training data.

2.1 Reasoning

Reasoning may be loosely defined as the process of drawing conclusions about unknown information based on what is already known. Contemporary dual process theory would modify this definition to emphasise conscious inference, distinguishing reasoning from unconscious processes like intuition [WE74; Kah03]. Some scholars challenge the conscious-unconscious distinction [MS18] and others propose alternative frameworks entirely [SW86; GCT24, inter alia]. Rather than adjudicating between these theoretical positions, this thesis adopts the broader definition outlined above for pragmatic reasons. Whether AI systems engage in conscious processing remains an open — and perhaps unanswerable — question. More immediately relevant is distinguishing reasoning from the retrieval of memorised patterns from training data. The distinction between a model that simply reproduces previously encountered information and one capable of generating genuinely novel knowledge is crucial for understanding AI capabilities. Under this definition, reasoning encompasses both deliberate logical inference and intuitive processes, making it functionally equivalent to 'inference' more broadly. I retain the term 'reasoning' because the tasks examined in this thesis are conventionally classified as reasoning problems when performed by humans. While I do not claim to definitively resolve whether current AI systems can reason under this definition, I aim to advance our understanding of the question.

2.1.1 Social Reasoning

Humans are pre-eminently social beings. Our capacity for social reasoning is thought to be intimately linked and sometimes even causally related to cognition [Tom99; VYG78; MAD07]. More directly, social reasoning enables us to navigate interpersonal interactions: understanding the mental states of others, inferring communicative intentions, and recognising the implicit rules that govern social behaviour. This multifaceted capacity develops through years of embodied social experience, beginning in infancy when children learn to follow gaze, respond to pointing, and engage in joint attention [TCL07]. By early childhood, humans demonstrate sophisticated abilities to understand that others may hold beliefs different from their own, anticipate how social context shapes meaning, and navigate the unspoken conventions that structure communication [reviewed in Wel90].

In this thesis, I investigate two components of social reasoning that are particularly relevant to language understanding. Firstly, pragmatic reasoning enables us to derive meaning that goes beyond the literal content of utterances, drawing on context, beliefs, and social institutions [Wit53; Gri75; Hua17]. A well-studied form of pragmatic language is conversational implicature, utterances that convey something other than their literal meaning. Consider an exchange where Esther asks her friend Juan "Can you come to my party on Friday?" and Juan responds "I have to work". We resolve Juan's response as him declining the invitation by using the contextual common-sense knowledge that having to work on a Friday night precludes attendance. This is an example of a conversational implicature, illustrating how unstated context contributes to meaning. In Appendix D.1, I present a brief literature review on implicature, covering the Gricean cooperative principle as well as other theories of pragmatics.

Secondly, theory of mind represents our ability to attribute mental states to others and understand how these states guide behaviour. This capacity follows a developmental trajectory beginning with basic goal attribution in infancy. Woodward [Woo98] demonstrates that 6- and 9-month-old infants can distinguish goal-directed from non-goal-directed actions, selectively encoding aspects of human actions that are relevant to the actor's goal object rather than other salient features of the event. Based on agents' reaching behaviour, infants form expectations about their object preferences, suggesting early understanding of goal-directed action. Studies of pre-linguistic infants such as these rely heavily on the looking-time paradigm, which exploits infants' tendency to look longer at novel or unexpected stimuli [Fan64]. This methodology has revolutionised developmental psychology, allowing researchers to build theories of infant cognition. In this paradigm, researchers first familiarise groups of infants with stimuli until their looking time decreases (i.e. *habituation*), indicating reduced interest. Subsequently, different test stimuli are presented to different groups of infants, and looking times are compared across conditions. When infants look longer at one test stimulus than another, this reveals how they represented and processed the original habituation stimuli, providing insights into early cognitive abilities.

Woodward [Woo98] uses the looking-time paradigm to study how infants represent goal-direct reaching motions. She habituates infants to reaching actions of a demonstrator that always reaches to the same object on the same location (e.g. a teddy bear on the left) over another object in another location (a ball on the right). The objects then switch positions, and infants looking time in two different test conditions is compared. In one condition, the actor reaches to the same object from habituation that is now in a different location (the teddy bear on the right), which would represent an object bias. In the other condition, the actor reaches to the other object in the same location from habituation (the ball on the left), demonstrating a location bias. Infants look longer for the location bias case, suggesting that this condition is more unexpected to them, which in turn suggests they expect the actor to reach for the teddy bear regardless of location. Woodward interprets this to mean they selectively encode the goal object of the actor's reach and not the location. Moreover, they do not show this behaviour when the actor is replaced by an inanimate rod that is moved to the object (the infants only see the rod and not whatever moves it). When they are habituated with a rod, the looking times in the object and location bias test cases are comparable.

Theory of mind abilities become increasingly sophisticated throughout childhood, culminating in the understanding of false beliefs: the recognition that others can hold beliefs about the world that differ from reality and from one's own knowledge [WP83]. Classic false-belief tasks reveal that by age 4–5, children understand that someone who has not witnessed a change in object location will continue to believe the object remains in its original position, even when this belief is objectively false. This developmental progression from goal attribution to false-belief understanding reflects the gradual emergence of our capacity to model the complex mental worlds of other agents.

2.2 Large Language Models

Language models came to be colloquially considered 'large' around GPT-3 [Bro+20a], but their foundational principles have deeper historical roots. The core self-supervised objective that drives both pre-training and types of post-training, next-token prediction, was pioneered decades earlier by Elman [Elm90] in his seminal work on recurrent neural networks. Elman demonstrates that by implicitly representing 'time' (i.e. temporal information) through its effect on processing (by maintaining a dynamic state), simple predictive objectives enable models to discover sophisticated linguistic structures, including word boundaries in character sequences and context-dependent lexical categories emerging from word order patterns. Beyond next-token prediction, many other foundational techniques in contemporary LLMs trace back longer, including dense word representations [Ben+03], sub-word tokenisation methods [Boj+17], and attention mechanisms for contextual processing [BCB15; LPM15]. In the following, I will briefly describe the stages of LLM development that are relevant for this thesis' contributions: pre-training, supervised post-training, and post-training with reinforcement learning. I will end with a note on the main approach driving progress and how it impacts the research presented here: scaling compute.

2.2.1 Pre-training

Nearly three decades after Elman's foundational study, Radford et al. [Rad+18] demonstrate that next-token prediction at scale can produce models that serve as powerful general-purpose representations, readily adaptable to diverse downstream tasks through fine-tuning. This breakthrough establishes the pre-training paradigm that now dominates natural language processing. Building on this foundation, Radford et al. [Rad+19] make another significant discovery: when scaling up the parameters and training set size of base generative models, they learn to perform zero-shot task generalisation directly from natural language *prompts*, without any fine-tuning. Specifically, a task can be specified by conditioning the model on a prompt $y_p = (y_{t_1}, \ldots, y_{t_m})$, and the model performs the task by generating a completion $y_c = (y_{t_{m+1}}, \ldots, y_{t_n})$, where $y_{t_i} \in \mathbb{Z}, \quad 0 \leq y_{t_i} < V$ are tokens from the model's vocabulary. Radford et al. validate this across multiple benchmarks spanning reading comprehension, translation, summarization, and question answering. Brown et al. [Bro+20a] subsequently show that this generalisation can be further enhanced by providing task examples within the input context itself, a technique now known as in-context learning. This capability has since spawned an active research field, with investigators exploring both the origins of in-context learning [Cha+22; Wur+25], what algorithm it implements [Aky+23; Von+23; ZFM25], and its practical applications for improving downstream task performance [Wei+22a]. In this thesis, we call generative pre-trained models simply *base models*, denoted by $\mathcal{M}_{\text{base}}$. We denote the model parameters by $\theta \in \mathbb{R}^D$, where D in this thesis ranges from millions (M) to billions (B) of parameters, and the distribution over next token it parametrises is denoted by:

$$p_\theta(y_c \mid y_p) = \prod_{i=m+1}^{n} p_\theta(y_{t_i} \mid y_{t_{<i}})$$

The parameters are typically found through next-token prediction using the negative log-likelihood:

$$\theta^\star = \arg\min_{\theta \in \mathbb{R}^D} \frac{1}{N} \sum_{x \in \mathcal{D}} \sum_{i=1}^{k} -\log p_\theta\big(x_{t_i} \mid x_{t_{<i}}\big) \tag{2.1}$$

$$\tag{2.2}$$

Where N is the training set size $|\mathcal{D}|$ and $x = (x_{t_1}, \ldots, x_{t_k}) \in \mathcal{D}$ training sequences of tokens $x_{t_i} \in \mathbb{Z}$. The parameters are typically found by performing a gradient-based iterative algorithm on the above objective that is stopped according to some criterion, which need not mean they minimise the objective or are converged.

2.2.2 Post-training

Pre-training enables LLMs to develop broad capabilities across diverse tasks, but base models often fail to align with user intentions. When prompted with "explain the moon landing to a 6 year old in a few sentences," base GPT-3 completes this with similar instruction patterns like "explain the theory of gravity to a 6 year old" rather than actually explaining the moon landing [Ouy+22]. To adress this, Ouyang et al. [Ouy+22] introduce a multiple stage post-training method. The first stage involves supervised fine-tuning (SFT), where the model learns from human-written instruction-completion pairs using standard next-token prediction. In subsequent stages, human evaluators rank multiple model outputs for various prompts, generating preference data that can be used to train the model through reinforcement learning from human feedback (RLHF). The resulting model is more aligned with human intent and vastly preferred by human evaluators, now completing the above prompt with "people went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them."

The alignment techniques pioneered by Ouyang et al. spurred extensive research into post-training methods. The field now encompasses a variety of training approaches and specialised instruction datasets spanning multiple domains. This thesis uses several of these advances: in Chapter 4, we incorporate MathInstruct [Tos+24] alongside our task-specific data when fine-tuning instruction-tuned models for code execution. This practice of mixing general instruction data with specialised training has become standard for preventing forgetting of instruction-following capabilities during domain-specific fine-tuning. The following outlines the three post-training methods used in this thesis: supervised fine-tuning (SFT), direct preference optimisation (DPO) [Raf+23], and group relative policy optimisation (GRPO) [Sha+24].

Supervised fine-tuning (SFT) is the most straightforward approach to adapting base models for specific behaviours, simply applying only the first stage of the alignment technique described above [Ouy+22]. SFT involves collecting relevant data and continuing model training with next-token prediction. Compared to pre-training, SFT typically requires substantially less data and computational resources. Research indicates this stage serves primarily to enhance existing capabilities acquired during pre-training rather than teaching fundamentally new skills [Jai+24; KSR24; Pra+24]. However, a growing body of work demonstrates that SFT has limitations in generalisation. Besides showing improved performance on top of SFT [Ouy+22], evidence suggests that models fine-tuned with reinforcement learning generalise better out-of-distribution [Kir+24; Chu+25; Sha+24].

Reinforcement learning from human feedback (RLHF), as described earlier, represents one prominent RL-based approach. In RLHF, a reward model is first trained using human preference data and then used to fine-tune the 'policy' (i.e. the model, typically following an SFT stage) via reinforcement learning objectives such as proximal policy optimisation (PPO) [Sch+17]. However, RLHF with PPO can be unstable and difficult to implement in practice. Rafailov et al. [Raf+23] propose a simpler alternative called **direct preference optimisation (DPO)**, which reformulates the two-stage RLHF process into a single-stage supervised training objective. The key insight in DPO is that, given a human preference pair ($y_{c,w}$ is preferred over $y_{c,l}$ for a given prompt y_p), one can analytically infer the optimal policy consistent with those preferences, eliminating the need for an explicit reward model. Using this insight, the DPO objective derives a logistic regression loss starting from the full RLHF objective. In the resulting objective, the logit is the difference in log-probabilities between the preferred and dispreferred response, and the target is 1 for the preferred sample, and 0 for the non-preferred one.

While DPO eliminates many of RLHF's implementation challenges, the field continues to develop more specialised approaches for particular domains. **Group relative policy optimisation (GRPO)** [Sha+24] is one of those methods, targeting mathematical reasoning tasks while significantly reducing PPO's computational overhead. The efficiency gains stem from eliminating PPO's value function requirement. As an actor-critic method, PPO must train a separate value function to estimate expected future rewards from each state. This is typically implemented with a copy of the base model, effectively doubling memory requirements. GRPO circumvents this by using a simpler baseline to estimate expected reward: the average reward across multiple sampled responses to the same prompt (where G represents the group size). This group-based baseline serves the same reward variance reduction purpose as a learned value function but requires no additional parameters. GRPO supports both outcome supervision, where rewards are assigned only at sequence completion, and process supervision, where intermediate rewards guide reasoning at each step. This flexibility makes it particularly well-suited for mathematical domains where step-by-step verification is valuable.

A key distinction among RL-based post-training methods is whether they operate on-policy or off-policy. On-policy methods learn from data generated by the current policy being optimised, requiring fresh samples at each training step. Off-policy methods can learn from data generated by any policy, including older model versions or entirely different models. This enables efficient reuse of existing datasets. DPO operates off-policy, allowing training on static preference datasets without generating new model responses during optimisation. GRPO and RLHF are on-policy methods that sample fresh responses from the current policy at each update step. However, RLHF introduces complexity through its reward model, which is typically trained on preference data collected from an earlier policy (usually post-SFT), creating a subtle distribution mismatch. These distinctions have practical implications for generalisation. In Chapter 4, we contribute to the literature demonstrating that RL has generalisation benefits, demonstrating that GRPO outperforms both DPO and SFT in generalisation. This suggests that the combination of negative sampling and on-policy training provides benefits for generalisation.

We denote post-trained models by $\mathcal{M}_{\text{stage}}$, where `stage` can be replaced by `SFT`, `GRPO`, or `DPO`, to denote which post-training method is used.

2.2.3 Compute scaling

The unifying perspective that abstracts away from the above underlying methodological details is provided by scaling laws [Kap+20; Hof+22]. These empirically-derived power laws describe how model loss decreases predictably as parameter count and dataset size increase. Crucially, scaling laws suggest that performance gains can continue indefinitely through optimal allocation of computational resources (ensuring neither resource becomes a bottleneck), and reveal that compute has been the primary catalyst for transforming these decades-old foundational principles into today's capable systems. See Figure 2.1 for an example of a scaling law.

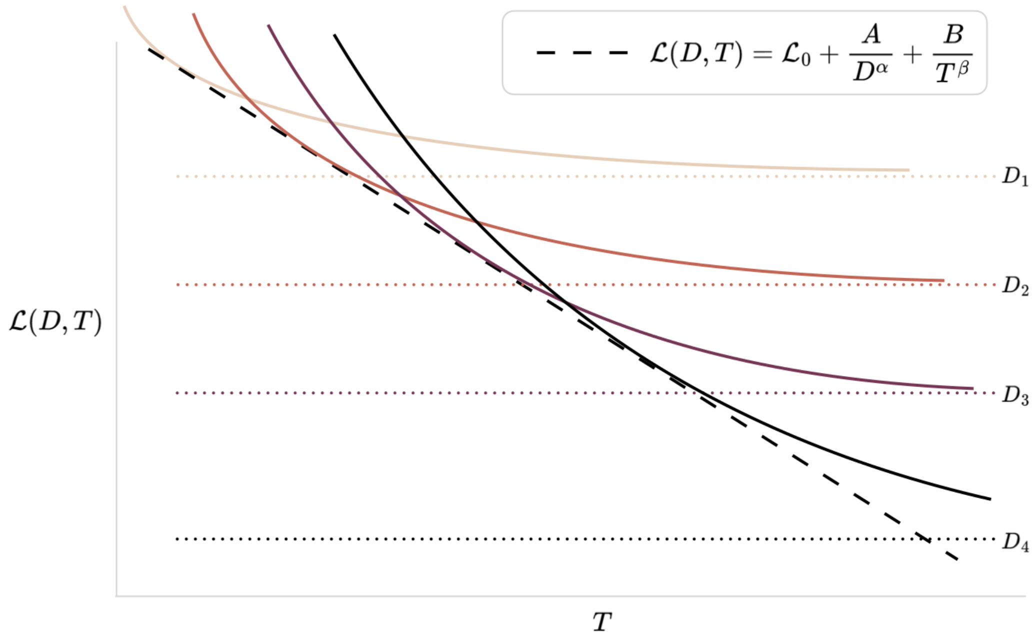


Figure 2.1: A Chinchilla scaling law [Hof+22], where T is the number of tokens, and $\mathcal{L}(D,T)$ is the estimated loss of a model with D parameters and T tokens. The parameters $\mathcal{L}_0$, A, B, α and β are empirically fitted across multiple model and token sizes (shown here for four model sizes D_1 to D_4). Using Chinchilla scaling laws, the compute-optimal model and token size can be obtained by minimising $\mathcal{L}(D,T)$ subject to compute constraints.

The scaling paradigm has two critical implications for the research in this thesis. First, the massive scale of training data (ranging from billions to trillions of tokens) makes it impossible to guarantee clean separation between training and evaluation data. Each chapter addresses this challenge differently: explicitly linking model responses to training data (Chapter 3), designing tasks unlikely to appear in pre-training (Chapter 4), testing for memorisation of evaluation data (Chapter 5), or employing control tasks (Chapter 6). Second, any observed effect or capability must be examined across different model and data scales, as phenomena that appear robust at one scale may diminish or disappear entirely as models or datasets grow larger.

2.3 Influence Functions

In response to the scaling paradigm, a field that is growing in importance is training data attribution (TDA): tracing back model behaviour to the training data. TDA aims to estimate the impact of data points on the trained model parameters and outputs. A naive way to calculate the influence of a particular training data point $x = (x_{t_1}, \dots, x_{t_k})$ (called 'document' in the context of TDA) on a model completion y_c given a prompt y_p (called 'query') is the following: train a model with x and without x in the training set $\mathcal{D}$, resulting in parameters $\theta_{\{x\}}$ and $\theta_{\{x\}^c}$ respectively. The influence of x is then given by the difference in likelihood assigned by the models to the completion given the prompt: $p_{\theta_{\{x\}}}(y_c \mid y_p) - p_{\theta_{\{x\}^c}}(y_c \mid y_p)$. However, this so-called leave-one-out (LOO) retraining is intractable for large language models.

In this section, I present the relevant background material for understanding the method used in Chapter 3 for tracing model behaviour back to pre-training data. I start by demonstrating in detail how Koh and Liang [KL17] estimate influence by repurposing a tool from robust statistics called influence functions [Ham74]. Then, I discuss how computational instabilities lead to changes in the influence function formulation [Tes+21], which in turns leads to them estimating a slightly different ground-truth than LOO retraining [Bae+22]. Finally, I briefly cover efficient estimation of the second-order terms in influence functions, which is required for modern-scale LLMs.

Classical influence functions. To estimate the influence of documents on queries, Koh and Liang [KL17] propose to use influence functions [Ham74]. Recall we want to calculate the influence of document x on the likelihood the model assigns to a query $p_\theta(y_c \mid y_p)$. Let us consider how Koh and Liang do this for the optimal parameters $\theta^\star$ that minimise the empirical risk $\mathcal{J}(\theta, \mathcal{D}) \coloneqq \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(x_i, \theta)$ (where $\mathcal{L}(x, \theta)$ is some scalar-valued loss function). We assume throughout that

1. $\mathcal{J}(\theta)$ and $\mathcal{L}(x, \theta)$ are twice continuously differentiable in a neighbourhood of $\theta^\star$,
2. the Hessian $H_{\theta^\star} \coloneqq \nabla^2 \mathcal{J}(\theta^\star)$ is non-singular (invertible).

Influence functions compute the change in optimal parameters if x were up-weighted by some small ε:

$$\theta^\star_{\varepsilon,x} \coloneqq \arg\min_{\theta \in \mathbb{R}^D} \left(\mathcal{J}(\theta, \mathcal{D}) + \varepsilon \mathcal{L}(x, \theta)\right) \tag{2.3}$$

Equation 2.3 is called a response function. We want to know how $\theta^\star_{\varepsilon,x}$ changes with ε, described by $\left.\frac{d\theta^\star_{\varepsilon,x}}{d\varepsilon}\right|_{\varepsilon=0}$. Since $\theta^\star_{\varepsilon,x}$ is defined by a minimisation, we need to use implicit differentiation using the first-order optimality condition (FOC) to obtain $\left.\frac{d\theta^\star_{\varepsilon,x}}{d\varepsilon}\right|_{\varepsilon=0}$. This is possible to do directly, as shown in Appendix B in [Bae+22]. Here, we instead follow the approach by Koh and Liang [KL17] who first take a Taylor expansion of the FOC, leading to the same result in a slightly more intuitive way. Given $\theta^\star_{\varepsilon,x}$ is a minimiser of $\mathcal{J}(\theta, \mathcal{D}) + \varepsilon\mathcal{L}(x, \theta)$, the FOC is:

$$0 = \nabla\mathcal{J}(\theta^\star_\varepsilon) + \varepsilon\nabla\mathcal{L}(x, \theta^\star_\varepsilon)$$

Where we drop the dependence of the empirical risk on $\mathcal{D}$ and of $\theta^\star_{\varepsilon,x}$ on x to avoid clutter, and we write ∇ in place of ∇_θ to further lighten notation (all gradients are w.r.t. θ unless otherwise specified). Now that we have an explicit expression, and since $\theta^\star_\varepsilon \to \theta^\star$ as $\varepsilon \to 0$, we can use a first-order Taylor expansion of the FOC around $\theta^\star$ to reason about how the optimal parameters may change when up-weighting x by ε.

$$0 \approx \left[\nabla\mathcal{J}(\theta^\star) + \varepsilon\nabla\mathcal{L}(x, \theta^\star)\right] + \left[\nabla^2\mathcal{J}(\theta^\star) + \varepsilon\nabla^2\mathcal{L}(x, \theta^\star)\right](\theta^\star_\varepsilon - \theta^\star) \tag{2.4}$$

How $\theta^\star_\varepsilon$ changes with ε (i.e. the quantity we are after) is the same as asking how $\theta^\star_\varepsilon - \theta^\star$ changes with ε, as $\theta^\star$ does not depend on ε. Therefore, we rewrite Equation 2.4 in terms of $\theta^\star_\varepsilon - \theta^\star$:

$$\theta^\star_\varepsilon - \theta^\star \approx -\left[\nabla^2\mathcal{J}(\theta^\star) + \varepsilon\nabla^2\mathcal{L}(x, \theta^\star)\right]^{-1}\left[\nabla\mathcal{J}(\theta^\star) + \varepsilon\nabla\mathcal{L}(x, \theta^\star)\right]$$

We further note that $\nabla\mathcal{J}(\theta^\star) = 0$, use the Woodbury matrix identity to expand the inverse, and only keep $\mathcal{O}(\varepsilon)$ terms:

$$\theta^\star_\varepsilon - \theta^\star \approx -\nabla^2\mathcal{J}(\theta^\star)^{-1}\nabla\mathcal{L}(x, \theta^\star)\varepsilon$$

Now, we take the derivative w.r.t. ε to obtain the influence of x on $\theta^\star$ as defined by influence functions:

$$\mathcal{I}_{\theta^\star}(x) := \left.\frac{d\theta^\star_\varepsilon}{d\varepsilon}\right|_{\varepsilon=0} \approx -\nabla^2 \mathcal{J}(\theta^\star)^{-1} \nabla \mathcal{L}(x, \theta^\star) = -H_{\theta^\star}^{-1} \nabla \mathcal{L}(x, \theta^\star)$$

An application of the chain rule then leads to the influence on the completion y_c given a prompt y_p as measured by any continuously differentiable quantity $f(\theta^\star)$ such as the loss or the likelihood $p_{\theta^\star}(y_c \mid y_p)$.

$$\mathcal{I}_{f(\theta^\star)}(x) = \left.\frac{df(\theta^\star_\varepsilon)}{d\varepsilon}\right|_{\varepsilon=0} = \nabla f(\theta^\star_\varepsilon) \left.\frac{d\theta^\star_\varepsilon}{d\varepsilon}\right|_{\varepsilon=0} = -\nabla f(\theta^\star)^T H_{\theta^\star}^{-1} \nabla \mathcal{L}(x, \theta^\star)$$

To provide an intuitive sense of what influence computed in this way measures, I discuss each term below. I refer to $H_{\theta^\star}^{-1} \nabla \mathcal{L}(x, \theta^\star)$ as the inverse-Hessian-conditioned document gradient and $\nabla f(\theta^\star)$ as the query gradient.

- $\nabla \mathcal{L}(x, \theta^\star) \in \mathbb{R}^{D \times 1}$: the sensitivity of the model's loss on x to small parameter changes, indicating the first-order direction that up-weighting document x would push the model parameters (in the opposite direction of the greatest increase in the loss given by the gradient).

- $H_{\theta^\star}^{-1} \in \mathbb{R}^{D \times D}$: modulates the effective magnitude of parameter changes when x is up-weighted based on the loss landscape's curvature at $\theta^\star$. Because of the inverse, document gradients aligning with low-curvature directions are amplified (flat directions on the loss surface), while those aligning with high-curvature directions are suppressed (sharp directions). Intuitively, during optimisation model parameters move more easily in flat regions of the loss than on sharp regions, meaning up-weighting x will be less "resisted" by the overall training dataset and lead to a higher influence of x on the parameters.

- $\nabla f(\theta^\star) \in \mathbb{R}^{1 \times D}$: the sensitivity of the target quantity f (evaluated on the query) to small parameter changes. Influence is the dot product between this query gradient and the inverse-Hessian-conditioned document gradient from the previous terms. If these two align in direction, then because of the leading minus sign influence is negative (indicating a decrease in loss or increase in likelihood when up-weighting x), and vice-versa. Intuitively, the interpretation on choosing f as the loss or likelihood is reversed because the loss equals the negative likelihood. The dot product encodes both directional alignment and magnitude; if either the query gradient or the Hessian-conditioned document gradient is close to zero, the influence will be small even if their directions align.

The above intuition demonstrates why introducing the expensive second-order Hessian term over first-order TDA methods such as TracIn [Pru+20] has benefits. The influence is not only determined by the document's gradient-similarity to the query, but also by how its gradient direction interacts with the global geometry of the loss surface shaped by the entire dataset. In Appendix B.1, we empirically demonstrate benefits of second-order information for data attribution in large language model fine-tuning.

Computational instabilities. Classical influence functions assume that the empirical risk is strictly convex in θ, ensuring a unique minimiser and an invertible Hessian H. However, this assumption fails in overparameterised neural networks, where many parameter vectors can achieve the same loss, leading to singular or nearly-singular Hessians. Basu, Pope, and Feizi [BPF21] and Bae et al. [Bae+22] empirically demonstrate that classical influence functions poorly approximate leave-one-out (LOO) retraining, tracing this failure to two core issues: non-unique optima due to overparameterisation, and the unrealistic assumption that models are trained to convergence. To address computational instability, researchers have proposed replacing the Hessian with more stable approximations. Koh and Liang [KL17] add damping terms $(H + \lambda I)$, while Teso et al. [Tes+21] use the Fisher Information Matrix (FIM) instead of the Hessian, which is positive semi-definite by construction. However, using the FIM comes at the cost of ignoring negative eigenvalues present in the true Hessian, corresponding to directions of negative curvature in the loss landscape. This omission can bias influence estimates by overestimating curvature magnitudes. Damping terms do not correct this bias, but they mitigate a separate numerical issue: instability when inverting nearly singular curvature matrices. For common loss functions like cross-entropy, the FIM is equivalent to the Gauss-Newton matrix $G = J^T H_{\hat{y}} J$, where J is the Jacobian of network outputs with respect to parameters and $H_{\hat{y}}$ is the Hessian of the loss with respect to outputs.

The true ground-truth estimated by influence functions for unconverged parameters. Bae et al. [Bae+22] argue that these modifications change what influence functions actually estimate. Rather than approximating

LOO retraining, damped influence functions with FIM/Gauss-Newton approximations actually estimate a different counterfactual: the proximal Bregman response function (PBRF). The PBRF measures how removing a data point affects model predictions while maintaining consistency with the original trained model, without requiring training to convergence. Starting from the proximal Bregman objective rather than classical empirical risk minimisation and using a linearisation of the PBRF, influence functions take the form:

$$\mathcal{I}_{f(\theta^u)}(x) = -\nabla f(\theta^u)^\top (G + \lambda I)^{-1} \nabla \mathcal{L}(x, \theta^u) \tag{2.5}$$

where θ^u represents the unconverged parameters from actual training, G is the Gauss-Newton matrix, and λ is the damping parameter. Bae et al. show that influence functions for modern neural network applications correlate better with the PBRF than LOO retraining. The PBRF formulation is therefore a more relevant counterfactual that accounts for the realities of modern neural network training. We adopt this PBRF-based formulation throughout our work, as it provides both computational stability and a more appropriate theoretical foundation for influence estimation in overparameterised models. For readers interested in the mathematical derivation connecting the PBRF to this influence function formulation, we refer to Appendices B.2 and B.3 of Bae et al. [Bae+22].

Efficient estimation of second-order information in influence functions. Computing Equation 2.5 requires an inverse-Hessian-vector product (IHVP). Following Grosse et al. [Gro+23], we use this terminology even though the computation involves the Gauss-Newton Hessian G (GNH) rather than the full Hessian H. While the IHVP can be computed more efficiently than explicitly forming G, it remains intractable for large language models with billions of parameters. To address this computational challenge, Grosse et al. [Gro+23] propose using eigenvalue-corrected Kronecker-factored approximate curvature (EKFAC) [Geo+18], which provides a tractable approximation to G based on the Kronecker-factored approximation to the curvature (K-FAC) method [MG15]. The FIM, equivalent to G in our case, is defined as follows:

$$F := \mathbb{E}_{x \sim p_{\text{data}}, \hat{y} \sim p_{\hat{y}|x}(\theta)}[\nabla \log p(\hat{y} \mid \theta, x) \nabla \log p(\hat{y} \mid \theta, x)^\top]$$

where p_{data} is the data distribution and $p_{\hat{y}|x}(\theta)$ is the model's predictive distribution. K-FAC approximates G by making two key independence assumptions:

the activations and pre-activations of each layer are independent, and different layers are independent of each other. These assumptions lead to a block-diagonal approximation where each block corresponds to a layer and can be factored as a Kronecker product, dramatically reducing computational complexity. EKFAC improves upon K-FAC by leveraging the insight that the Kronecker-factored blocks admit efficient eigendecomposition. This enables a more accurate approximation of G while maintaining computational tractability [Geo+18]. For detailed derivations of the EKFAC approximation, we refer readers to Grosse et al. [Gro+23], Section 2.2.2 and 2.2.3.

Chapter 3

How Models Learn to Reason from Pre-training Data

3.1 Overview

Having covered the topics needed to understand the content of this thesis, let us recall the overarching goal of this work: to characterise LLM reasoning in the context of the compute scaling paradigm. Most recent progress is driven by increased compute during development, with token budgets reaching trillions of tokens. At this scale, the boundary between training and evaluation data has effectively collapsed: we can no longer reasonably assume that benchmark problems are unseen during pre-training. This raises a question: to what extent does LLM reasoning depend on similar sequences encountered during pre-training? This question is particularly pressing given the conflicting evidence about LLM capabilities. Their well-documented versatile reasoning abilities [WHL23; WHL24; McL+24, inter alia] sharply contrast with the line of work highlighting the brittleness of their reasoning [Raz+22; McC+23; Ull23a; Wu+24; Mah+24]. A finding common to the latter is that LLM reasoning depends on the frequency of similar problems in the training data, meaning benchmark saturation on its own can not be taken at face value. Recent works have documented the extent of the contamination issue [Bro+20b; Tou+23; Gun+23; Yan+23; Den+24], showing that many common benchmarks have a high percentage of contaminated data. Yang et al. [Yan+23] show that even rephrased benchmark data that elude N-gram-based detection methods can impact performance, further complicating the issue. However, it is unclear how and when state-of-the-art LLMs rely on contaminated data to perform reasoning.

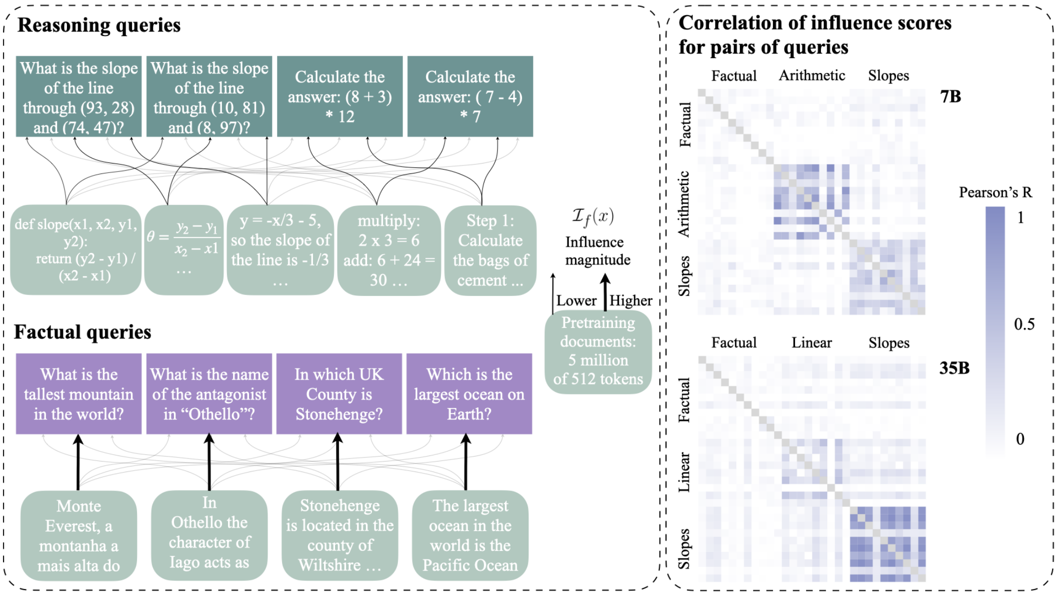


Figure 3.1: A summary of our most important findings towards answering the question *"how do LLMs learn to reason from pre-training data?"* We rank 5 million pre-training documents according to their influence on the likelihood of completions of two models, Cohere's Command R 7B and 35B, for 40 factual and 40 reasoning queries. We find that procedural knowledge drives influence on reasoning traces: a document's influence on the reasoning traces of one query is strongly predictive of that document's influence on another query with the same mathematical task, in 3 of the 4 tasks. We show this on the left through arrows indicating influence, and on the right through correlations of all 5M document influences between a random sample of 10 queries per task (a plot with all queries can be found in Figure B.8 in Appendix B.9.1). Further, we find that the answers to factual queries often show up in the top 0.01% of pre-training documents (see text in bottom row of documents), but not for the reasoning questions. Finally, individual documents influence reasoning traces much less strongly than factual answer generations, indicating models rely on documents less when reasoning. All documents and queries shown are redacted versions of real data, and the relations are based on documents found in the top 50 for the queries.

In this chapter, we investigate which pre-training data influence model's reasoning traces and how those data relate to the specific problems being addressed, contrasting it to the data that influence factual question answering. We use influence functions (as detailed in the previous Section 2.3) to compute the influence of pre-training documents on the likelihood of prompt-completions pairs under a trained model. In the extreme case, a language model answering reasoning questions may rely heavily on retrieval from parametric knowledge influenced by a limited set of documents within its pre-training data. In this scenario, specific documents containing the information to be retrieved (i.e. the reasoning traces) contribute significantly to the model's output, while many other documents play a minimal role. Conversely, at the other end of the spectrum, the model may draw from a broad range of documents that are more abstractly related to the question, with each document influencing many different questions similarly, but contributing a relatively small amount to the final output. We propose generalisable reasoning should look like the latter strategy.

Our findings, summarised in Figure 3.1, suggest a generalisation strategy for reasoning that is unlike retrieval from the parametric knowledge formed during pre-training. Instead, the models learn to apply procedural knowledge extracted from documents involving similar reasoning processes, either in the form of general descriptions of procedures, or applications of similar procedures.

3.2 Method

Given a pre-trained model $\theta^u \in \mathbb{R}^D$ that parametrises a distribution over next tokens conditioned on a prompt $p_{\theta^u}(y_c \mid y_p)$, we are interested in finding data from the pre-training set $\mathcal{D} = \{x_i\}_{i=1}^N$ that influence the completion. Put differently, we want to know which examples in the pre-training set 'caused' a completion. To this end, we use EKFAC influence functions for large-scale transformers as proposed by Grosse et al. [Gro+23]. Recall that the influence of a training document $x \in \mathcal{D}$ on a continuous differentiable function f of the parameters θ^u is given by Equation 2.5 in Section 2.3, copied here:

$$\mathcal{I}_{f(\theta^u)}(x) = -\nabla f(\theta^u)^\intercal (G + \lambda I)^{-1} \nabla \mathcal{L}(x, \theta^u)$$

Since we are investigating models with billions of parameters D, computing G is intractable, and we estimate it using EKFAC estimation (introduced in Section 2.3). To make this estimation tractable we make a number of simplifying assumptions across all our estimations, like independence between layers and we only take into account MLP parameters of the transformer layers [Gro+23]. A full list of approximations can be found in Appendix B.7.

What we measure influence on (i.e. choosing f). Prior work has shown that EKFAC influence functions more accurately estimate the counterfactual given by the response function (Equation 2.3) than other types of influence functions [Gro+23]. However, besides influence on language model completions, we are also interested in influence on the *accuracy* of a trained language model when answering questions. We can only calculate the influence on a continuous differentiable function, and to the best of our knowledge, no work has shown that influence functions also estimate effect on the underlying accuracy of text produced by next-token prediction. As a proxy for accuracy, we take as a continuous differentiable function the cross-entropy loss function (f in Equation 2.5). In Appendix B.1 we show that the influence calculated in this way surfaces documents that have a causal effect on the accuracy of a 7B model fine-tuned to do reasoning and reading comprehension tasks. Namely, if we remove documents from the fine-tuning data according to their influence and re-train the model, the accuracy drops significantly more than if we take out the same amount of documents randomly, or the same amount of documents using gradient similarity. In parallel, we motivate the use of EKFAC estimations of the GNH, by showing it significantly improves over a method using only first-order information.

Making the computation tractable. Besides the EKFAC estimation of G (for which we mostly follow prior work [Gro+23] and describe deviations in Appendix B.2), each influence score requires a document and query gradient. This means that if we would compute influence for the entire pre-training set it would be at least B times more expensive than pre-training the model itself (where B is the batch size). For this reason, we sample documents i.i.d. from the pre-training distribution. Still, it is only reasonably possible to loop over this sample once, and to store more than a single query gradient in memory (which has the same memory complexity as the model itself), Grosse et al. [Gro+23] use singular-value decomposition (SVD). Instead of SVD, we use approximate SVD with a probabilistic algorithm [HMT11], which significantly speeds up the computation of the query gradients. We justify each approximation in Appendix B.2.1.

Interpreting the influence score. We approximate Equation 2.5 to get scores for documents from the pre-training data $\mathcal{D}$ that represent the influence they have on a completion y_c given a prompt y_p. Given the counterfactual question approximated by the response function used to derive influence functions, an influence score of -1 for x implies the loss of the sequence y_c is decreased by

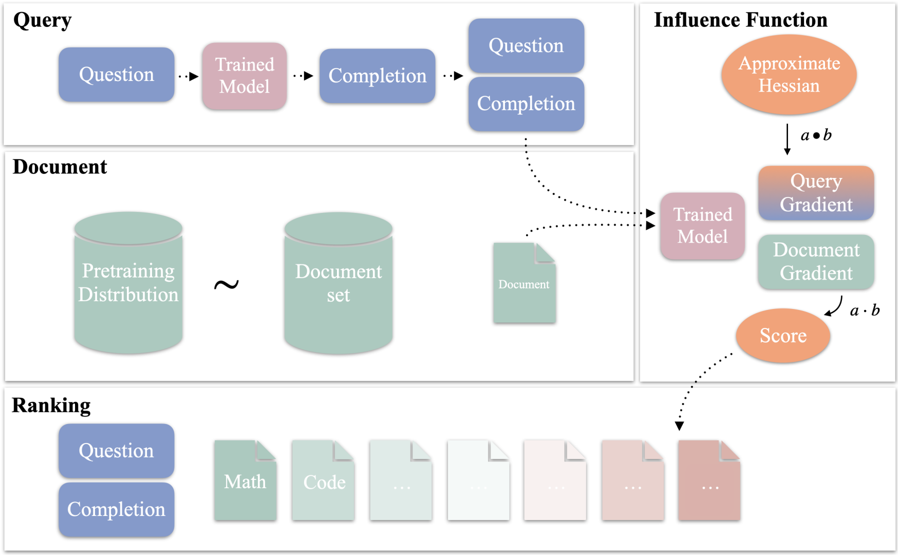


Figure 3.2: The pipeline for creating rankings of the most influential pre-training documents for a question-completion pair (*query*) using influence functions. The documents at the top of the ranking influence the likelihood of the completion positively, and the bottom negatively. We create rankings for a set of 40 reasoning, 40 factual, and 20 control queries over 5 million pre-training documents (2.5B tokens) for two models of different sizes (Cohere's Command R series, 7B and 35B).

1 due to its inclusion in the training set. To compare influence scores across different completions (and token lengths), we normalise the scores for each query by the information content of its completion y_c, measured in nats. The information content of a query is defined as $\mathbb{I}(y_c) = -\log\left(p_{\theta^u}(y_c \mid y_p)\right)$. The influence scores induce a ranking over documents from most positively to most negatively influential, where a score can be interpreted as the decrease (or increase) in loss per nat of query information. Note that we choose to use the terms 'positive' and 'negative' here colloquially to make the discussion of the results less confusing, where positive influence means a positive effect on the loss (by decreasing it). Similarly, in the rest of this chapter a positive total influence *decreases* the loss. The full pipeline is shown in Figure 3.2.

3.3 Experimental Setup

Query sets. We collect query sets with different types of questions, of which 40 are reasoning questions and 40 factual questions. Note that it is only tractable to loop over the pre-training sample we look at once, so we need to be able to store all query gradients in memory and cannot go beyond about 100 questions. For the remaining 20 queries that fit in memory, we design 10

Table 3.1: Example from the reasoning set that involves simple two-step arithmetic.

Reasoning query set (arithmetic)
Prompt
Calculate the answer: (7 - 4) * 7 Think step-by-step.
Completion (by 7B)
First, let's evaluate the expression inside the parentheses: 7 - 4 = 3 Now, let's multiply the result by 7: 3 * 7 = 21 Therefore, the answer is 21.

Table 3.2: Example from the factual set that requires retrieving the right answer.

Factual query set
Prompt
What is the tallest mountain in the world and how tall is it?
Completion (by 7B)
The tallest mountain in the world is Mount Everest, which is located in the Himalayas. It is 29,029 feet tall.

control questions for each query set (reasoning control and factual control). These control questions are superficially similar to reasoning or factual queries, but do not require reasoning or factual retrieval to be completed accurately (examples given in Appendix B.3). For the reasoning questions, we identify two types of mathematical reasoning each model can do robustly with zero-shot chain-of-thought [Wei+22b]. We do this by evaluating the models on larger sets of 100 questions for each type of reasoning, and selecting tasks where it gets at least 80% correct. This surfaces simple two-step arithmetic for the 7B model (Table 3.1), calculating the slope between two numbers for both models (of which two redacted examples are shown in Figure 3.1), and solving for x in linear equations for the 35B model (see Table B.7 in Appendix B.3 for prompt-completion pairs of the linear equations task). We ensure no query ever requires outputting a fraction. To make the results between 7B and 35B more comparable, we use the same slope questions for both models. For the 40 factual questions, we make sure the model gets half right and half wrong, allowing us to identify failures of retrieving facts from parametric knowledge, and we also ensure 16 of 40 overlap between models. We calculate influence over the full completion, which includes the chain-of-thought in the reasoning case. The full query sets are provided in the supplement.[1]

[1]Which can be found at `https://openreview.net/forum?id=1hQKHHUsMx`.

Documents set. We want to compare the influence of pre-training data on reasoning by differently sized models (7B and 35B), so we select two models that are trained on the same data. The EKFAC estimation of the GNH only needs to be done once per model, but the other terms in Equation 2.5 require two forward- and backward-passes through the model per document-query pair. This means that obtaining a ranking over pre-training data for a single query has a computational complexity similar to pre-training itself. To overcome this issue, we sample a set of documents from the pre-training data that covers multiple examples from each batch seen during pre-training, giving a total of 5 million documents (approximately 2.5B tokens) distributed similary as the training distribution. We batch queries and obtain the influence scores in parallel. Each document contains 512 tokens.[2]

EKFAC estimation. To estimate the GNH for the 7B and 35B models we need to estimate two expectations w.r.t. the data distribution (see Section 2.2.2 and 2.2.3 in [Gro+23]). To this end, we randomly sample 100 000 documents equally spread-out through pre-training for both models. Details on how exactly we approximate the GNH are in Appendix B.2. We note here that although this aspect of the pipeline requires estimating over 300B parameters representing second-order information, the bottleneck remains calculating document gradients.

Models. We look at two models of different sizes: 7B and 35B (Cohere's Command R series).[3] For each model, we use both base and supervised fine-tuned versions, where the former is trained on trillions of tokens and the latter is subsequently fine-tuned on a few thousand instruction-completion pairs. We estimate the second order information and calculate document gradients using the base models, and generate completions and calculate the query gradients using the models fine-tuned with supervised instruction-tuning. The reason for choosing this setup is that the fine-tuned models are much better at instruction following. This means we are assuming the EKFAC for the fine-tuning phase is the identity [Bae+24], and we are focusing only on the influence of the pre-training data and ignoring the fine-tuning data.

[2] We choose 512 tokens because qualitatively interpreting more is hard (usually spanning multiple topics).

[3] The specific stages of models we use are not publicly available, but the final checkpoints are: `https://huggingface.co/CohereLabs/c4ai-command-r-v01`.

3.4 Experiments and Results

We compare the rankings (from most positively to most negatively influential) over pre-training data produced by influence functions for reasoning questions to the rankings for factual questions (which can only be answered by retrieving parametric knowledge). We first analyse the rankings quantitatively by looking at the influence of different documents per nat of generated query information (Section 3.4.1). We aim to elucidate how generalisable the information in the influential documents is, and how many documents the model is relying on when doing reasoning compared to retrieval. Then, in Section 3.4.2 we investigate how the documents relate to the queries qualitatively.

3.4.1 Quantitative analysis

Finding 1: There is a significant positive correlation between the influence scores of documents for queries with the same underlying reasoning task, indicating that these documents are relevant for questions requiring the same procedure applied to different numbers. If models are relying on documents that contain 'general' knowledge that is applicable to any query with the same task (e.g. queries that require finding the slope between two points for many different points), we would expect there to be a significant correlation in the influence scores for these queries. We calculate the Pearson's R correlation of all 5 million document scores for all query combinations (leading to 100^2 correlations per model). The results can be seen in the right panel of Figure 3.1 for a subsample of 10 queries per task, and all query correlations can be found in Figure B.8 in Appendix B.9.1. We find a strongly significant (p-values all below $4e-8$) positive correlation between many queries of the same reasoning type, and a strongly significant absence of correlation (p-values all around $4e-3$) for most (but not all) factual queries or other combinations (e.g. reasoning queries of different types). This means that many documents have a similar influence on the same type of reasoning. Put differently; the model learns from the same data for different instances of a reasoning task. Given that each type of reasoning query requires applying the same procedure to different numbers, the positive correlation indicates that the influence scores for reasoning queries pick up on procedural knowledge. The correlations are strongest for the slope queries by the 35B model, and this is also the type of reasoning the model can do most robustly compared to solving linear equations. For the model to be able to solve linear equations with an accuracy of more than 80%, we restrict the calculations to lead to positive x, whereas for the slopes questions the answers can be positive or negative. In Appendix B.9.1 we falsify the hypothesis that the correlations are caused by the fact that the reasoning questions are superficially similar to each other, by using the set of control queries that is also superficially similar but does not require any reasoning and repeating the entire experiment. For the control queries we mostly do not observe a correlation. In Appendix B.9.1 we highlight examples of queries with high or low correlation for different query sets, finding that some of the correlation seems driven by formatting of reasoning steps, and most by reasoning procedure. For example, we find correlations are not driven by superficial similarities between queries like "think step-by-step" or the word "slope": when these phrases co-occur in control queries, there is usually not a significant correlation of their influence scores.

Finding 2: When reasoning, the model on average relies on each individual document less per generated nat of information than when answering factual questions, and the total magnitude of influence is less volatile, indicating it is generalising from a more general set of documents. The effect is more pronounced for the larger model.

In Figure 3.3 we show the total influence for different percentiles of the

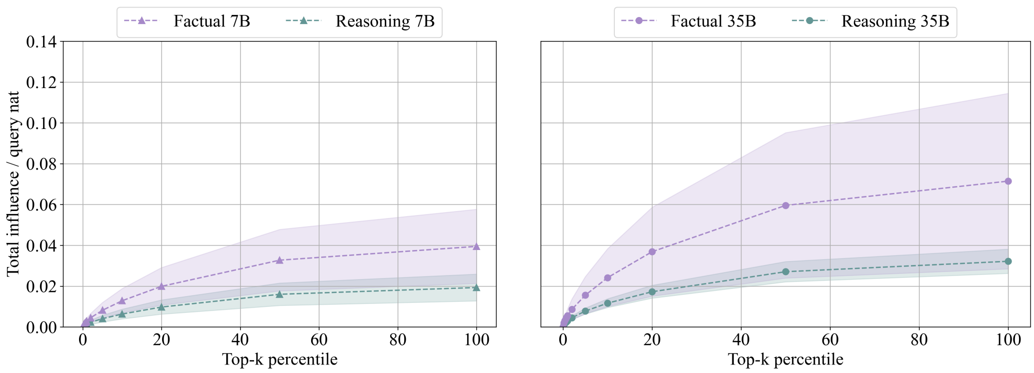


Figure 3.3: The total influence per nat of query completion information for different portions of the positive ranking over documents, left for the 7B model, right for the 35B. The total influence per nat is usually lower for reasoning questions than for factual questions, and the influence per document varies more for factual questions than for reasoning questions, especially for the 35B model.

positive parts of the rankings. The results depict the total amount of influence contained in the top-k percentile of the positively ranked documents: e.g. the

20th percentile contains 20% of the positive documents for a query, and the amount of total influence shown is the sum of all document influences up to that part of the ranking. The equivalent for the negative portions looks similar (Figure B.11 in Appendix B.9.2) and the discussion below applies similarly to the negative ranking. We observe two things for both models. Firstly, the amount of total influence for most factual questions at any part of the ranking is higher than for reasoning questions. Secondly, there is more variation in the influence of documents at the same rank across different factual queries (and for a few factual queries the amount of influence is actually lower than for the reasoning queries, seen more clearly in Figure B.16 in Appendix B.9.3). The first result means that, on average, the models rely on individual documents within our set less for generating reasoning traces than for answering factual questions. The second result indicates that for the factual questions the model relies on more 'specific' and infrequent documents: for a factual question it is more up to chance whether relatively highly influential documents (w.r.t. influence of documents for other factual questions) are part of the pre-training sample or not.

Influence spread. Another way to analyse the magnitude of influence is to look at the dispersion of influence across the ranking: how much of total influence for each query is contained at the top and bottom parts of the ranking? Similarly to what Grosse et al. [Gro+23] report, we observe that the top parts of the rankings over documents follow a power law characterised by a linear relation between rank and influence per nat in log-log space (shown in Figure B.16 in Appendix B.9.3). We find that the slopes for the reasoning questions by the 35B are slightly steeper than for the factual questions, and therefore the percentage of positive influence contained in the top portions of the rankings for the 35B reasoning questions increases faster with rank than for the factual questions (shown in Figure B.18 in Appendix B.9.3). For the 7B, the slopes for the reasoning questions the model gets right are on average also a bit steeper than for the factual questions, but the effect goes away when comparing slopes for all factual vs. reasoning queries. This means that the percentage of the total positive influence the top sequences cover is higher for the reasoning questions than for the factual questions for the 35B model (and similarly for the bottom sequences, see Figure B.11). There is a chance this finding is caused by noise for the 35B model and we discuss this possibility more in Appendix B.9.3, where we note that for the reasoning query with the steepest power law, the top 1 document is qualitatively entirely unrelated to the prompt.

If we compare the result between models, we find that the difference in magnitude and volatility are more pronounced for the 35B model across the full rankings. We look into this in Appendix B.9.2, and find that the effect remains even if we only look at queries that are the same for both models, which points to higher data efficiency for the larger model.

3.4.2 Qualitative analysis

We perform three qualitative analyses on the top portions of the rankings for each query; we search for the answer, we characterise the documents' relation to the reasoning queries, and we investigate what source datasets they are from (for both the top and bottom parts of the ranking, e.g. code, Wikipedia, etc). To filter some of the noise, we divide the influence scores by the document gradient norm and re-rank them, which has empirically been found to help [Cho+24].

Finding 3: The answer to the factual questions shows up relatively often in the top influential documents for the factual questions, and almost never for the reasoning questions.

To find the answer to the questions in the queries in the top documents

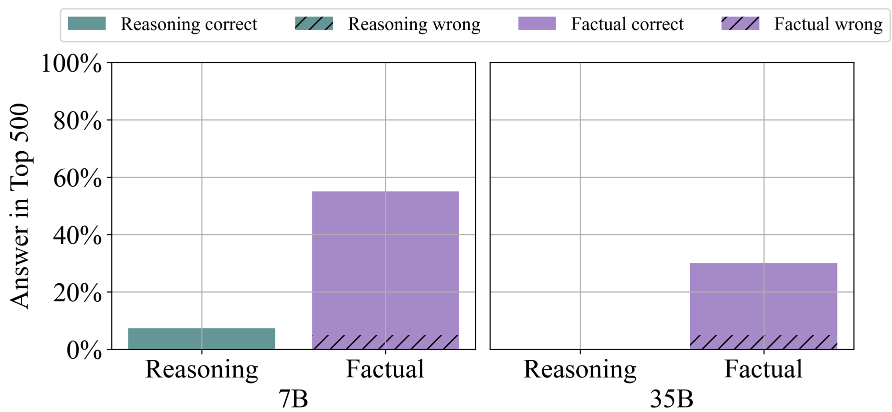


Figure 3.4: We search for the answer in the top 500 (top 0.01%) documents, and find it relatively frequently for the factual questions. For the reasoning questions, we find the answer twice for the 7B, and never for the 35B. Both those times, the answers to the steps occur in separate documents.

manually, we construct keywords for each query that should be in the document if the answer is there. For example, for the factual query in Table 3.2, the keywords are "tallest", "highest", "Mount Everest", "29029", "8848". For the reasoning queries, we construct many more keywords per query, but some examples for the example in Table 3.2 are $7-4$, 3, 21, $3*7$, as well as replacing the operations with words like ‘minus’ and ‘times’, and different ways of representing the content in this query. For details on which keywords we use for each query, see Appendix B.4. We determine the occurrence of each of these keywords independently in the top 100 documents for each query (meaning even if just the keyword ‘7’ is present it would be a hit), resulting in many false-positives. We manually look over the hits to find the answer. On top of that, we craft a prompt for Command R+ (a more capable 100B model) to find the answer in a query-document pair, and use it to find the answer in the top 500 documents for each query independent of keyword overlap (the prompt is given in Appendix B.5). Then, we manually look over the hits and keep track of documents that have the answer to a query. We verify that Command R+ finds all, and more, of the answers we have identified manually. We look for the full answer in a single document. For the reasoning queries, we also count partial answers in separate documents if they combine to the full answer. For example, if one document contains $7 - 4 = 3$, and another $3 * 7 = 21$, we consider that an answer. Finally, we apply the keyword overlap search combined with prompting Command R+ to a subset of the broader 2.5B pre-training tokens to verify that the answer to the questions are in the entire set even if they do not show up in the top 500 documents for queries.

The results are shown in Figure 3.4. For the 7B model, we find the answer in the top 500 documents for 55% of the factual queries, compared to 7.4% of the reasoning queries. For the 35B model, the answer to the factual queries shows up in the top influential documents 30% of the time, and never for the reasoning set. We expect the answer shows up less frequently for the 35B model simply because the factual questions are much more ‘niche’. For example, one of the questions the model gets correct is *“In which year did the Beinecke Library open?”*. Moreover, in certain cases, the answer shows up multiple times in the top 500 documents. If we count all these separately, as opposed to a binary ‘yes’ or ‘no’ per query on which the results in Figure 3.4 are based, answers to questions show up 30 times for the factual questions in the 7B rankings, and twice for the reasoning questions. For the 35B, the same result is 15 times for the factual questions, and never for the reasoning questions. Interestingly, the answer to the factual questions often shows up in different languages, like Spanish or Portuguese. We give two examples in Appendix B.8.2. To falsify the hypothesis that the answers to reasoning questions are not showing up because they are not present in the set of 5M documents, we repeat the above keyword search over a random subset of the 5M documents. We identify answers to reasoning steps in documents that do not show up in the top 500 documents for 13 of 20 arithmetic queries and a full answer for 1 of 20, and expect more to be there that elude the keyword search. For the slopes and linear equation queries, we find answers to 3 reasoning steps which do not show up in the top 0.01%. In Appendix B.8.1 we show some of these documents and their ranks.

Finding 4: We find that influential documents for the reasoning queries are often doing a similar form of step-by-step reasoning, e.g. also arithmetic. Further, we find that the influential documents often implement a solution to reasoning questions in code or general math.

For the slope queries (of which we have 20 which are the same for both models), many different documents surface as highly influential that show how to calculate the slope between two points in code or math. For the 7B model, documents that present explicit procedural knowledge on how to calculate the slope in either code or math show up in the top 100 documents for 16/20 queries (38 times), and for the 35B model they show up for all queries (51 times). All together, we manually find 7 unique documents that implement the slope in code in the top 100 documents, and 13 that present equations for calculating the slope. The 7B model relies on 18 of these documents for its completions (meaning 18 different ones appear in the top 100 documents for all queries), and the 35B on 8. An example of a highly influential document implementing the solution in JavaScript (left) and in maths (right):

Positively influential code

```
function eqOfLine(x1, y1, x2, y2) {
  if (x1 === x2) {
    // Handle a vertical line
    return `x = ${x1}`;
  } else {
    // Calculate the slope
    const m = (y2 - y1) / (x2 - x1);
    const b = y1 - m * x1;
    // Return y = mx + b
    return `y = ${m}x + ${b}`;
  }
}
```

Positively influential math

If a straight line passing through the points $P(x_1, y_1), Q(x_2, y_2)$ is making an angle θ with the positive X-axis, then the slope of the straight line is:

(A) $\frac{y_2+y_1}{x_2+x_1}$
(B) θ
(C) $\frac{y_2-y_1}{x_2-x_1}$
(D) $\sin\theta$

Solution:
Correct answer: (C)

We prompt Command R+ to further characterise the top 500 documents for each query by choosing from a set of provided keywords, and find that often the documents are doing similar arithmetic on other numbers (e.g. much larger or smaller), doing similar arithmetic on similar numbers (for the slope questions), or similar algebraic operations on similar numbers (for solving linear equations). We present the detailed results and prompt for this analysis in Appendix B.8.3.

Finding 5: For factual queries, the most influential data sources include Wikipedia and trivia, while for reasoning, key sources consist of maths, StackExchange, ArXiv, and code.

We look at the type of source datasets that represent the most influential documents. Specifically, we count the source datasets of the top and bottom k documents with $k \in \{50, 500, 5000, 50000, 500000\}$, and compare the count to the pre-training distribution. We present the details in Appendix B.8.4, but mention here that code data is highly influential for reasoning. StackExchange as a source has ten times more influential data in the top portions of the rankings than expected if the influential data was randomly sampled from the pre-training distribution. Other code sources are twice as influential as expected when drawing randomly from the pre-training distribution for $k = 50$ up to $k = 50000$. Similar patterns hold for the bottom portions of the rankings.

3.5 Related work

The subfield with the aim of understanding how large language models generalise is growing rapidly. This question can be approached in different ways, and many recent works interpret weights of smaller models on synthetic tasks to explain particular phenomena that we observe in language models at scale such as grokking [Wan+24a], in-context learning [Ols+22; Sin+24], or superposition [Elh+22; Bri+23]. Scaling interpretability methods to modern-sized LLMs is challenging for many reasons, of which one is computational tractability. Nonetheless, there are a few works that apply techniques from interpretability to language models at scale. Templeton et al. [Tem+24] use sparse autoencoders to extract interpretable features from Claude 3 Sonnet, and demonstrate how to use these features to control model outputs. Grosse et al. [Gro+23] adapt EKFAC influence functions [Geo+18] to large-scale Transformers, and use them to understand what kind of pre-training data influence completions of models up to 50B parameters. The authors show, among many other things, that larger models rely on pre-training data that are more abstractly related to the completion than smaller models. In this chapter, we build on the results of Grosse et al. [Gro+23], leaning heavily on their efforts to make influence functions tractable at scale, but focus instead on understanding reasoning specifically.

3.6 Discussion, Limitations, and Future Work

From the findings in this chapter, we conclude that the generalisation strategy for reasoning LLMs employ is unlike retrieval. More often than not, even if the answer is part of the set of pre-training documents we look at, it does not show up as highly influential as the answers to factual questions do. We find that instead, the positively influential documents often contain explicit procedural knowledge on how to get to a solution. Further, the models rely less on individual documents when reasoning than when answering factual questions, and the set of documents they rely on is more general. Finally, documents often have a similar influence on reasoning queries that require applying the same procedure to different numbers.

One of the quantitative findings is counter to our initial expectations: we find that the distribution of influence is less spread out for reasoning than for factual questions, characterised by steeper power laws. The distribution of influence over documents tells us something about the type of generalisation strategy the model is using; the more documents that contribute to each nat of query information (i.e. the more spread out the total influence), the more documents the model is relying on to produce the completion. One would perhaps expect a steeper power law for factual questions than for reasoning (meaning more of the total positive influence contained at the top parts of the ranking), but our results show evidence for the opposite. Perhaps a model needs to generalise from a broader set of documents for factual retrieval than for reasoning because it needs to see the same information more often to memorise it. This is supported by the finding that for factual questions the answer often shows up multiple times in the top 0.01% most influential data.

There are important limitations to our approach, most notably that we do not calculate influence on the entire training set, which is intractable. An alternative explanation of our results is then the opposite conclusion: the model is relying on data for reasoning that are so infrequent that a random sample of 2.5B tokens does not surface relatively highly influential samples for any of the 60 unique reasoning queries. This would result in the conclusion that LLMs rely on sparse and infrequent documents for reasoning. That means we are effectively looking at a set of relatively uninfluential documents for reasoning, and that perhaps the answers to reasoning traces would be highly influential when looking at the entire pre-training data. We would argue that this is the more unlikely explanation for three reasons: (1) the qualitative analysis shows that the influential data for the reasoning questions are intuitively highly relevant, and that the answers to many reasoning traces *are* part of the 2.5B tokens, they are just not highly influential for reasoning, (2) the correlation of influence scores for the different reasoning tasks is highly significant, and (3) we confirm that these results do not hold for control queries that look similar to the reasoning queries superficially, but do not require step-by-step reasoning. Moreover, it seems unlikely that the model is learning to do retrieval from such infrequent data for one of the simplest forms of mathematical reasoning, namely subtraction and multiplication on small numbers. Taken together we argue the results indicate a generalisation strategy that relies on procedural knowledge. Regardless, the nature of interpretability research such as the work presented here is that all we can do is provide evidence, and not proof.

Another limitation is that we do not look at the supervised fine-tuning stage. The reason we only look at the pre-training data is because the fine-tuning stage is targeted at making the models more aligned and 'instructable', and prior work has shown that SFT serves primarily to enhance existing model capabilities [Jai+24; KSR24; Pra+24]. Nonetheless, an interesting direction for future work is applying the same method used here to the fine-tuning data.

With this work, we do not claim to say contamination is not an issue, or that LLM reasoning is not brittle and reliant on pre-training statistics. What we demonstrate is that, in principle, it appears possible for LLMs to produce reasoning traces using a generalisation strategy that combines information from procedurally related documents, as opposed to doing a form of retrieval. This is not to say that there are no cases of LLM reasoning where the model is in fact doing retrieval, on the contrary, models can be overfit to contaminated data if it appears often enough in the training data.

The work in this chapter spurs further avenues for future work. Firstly, as previously discussed, identifying data types that are similarly influential across reasoning types could provide additional insight into data selection techniques for improved reasoning. Relatedly, what properties of code data makes it influential for reasoning? What kind is positively influential, and what kind negatively? Further, since we only take into account the feed-forward layers and treat the attention as fixed, an interesting avenue for future work would be to investigate how the relatively low magnitude of influence of pre-training data on feed-forward parameters for reasoning traces interacts with attention, connecting to a finding from literature that certain forms of reasoning happen in the attention heads [Ols+22]. Finally, in this work we investigate mathematical reasoning. Future work should verify whether similar results hold for other types of reasoning, such as inductive reasoning.

The central finding of this chapter is that LLMs depend on procedural knowledge in pre-training for reasoning, demonstrated quantitatively by showing how a pre-training document's influence for one reasoning query is predictive of its influence on another query with the same task. Qualitatively, we identify explicit symbolic procedures that frequently prove highly influential for reasoning performance (Finding 4). However, the latter does not tell us why these explicit procedures are useful. While influence functions reveal that models consistently draw on abstract code implementations and analytical formulas, they cannot determine at what level of abstraction models process and use these data. The next chapter investigates whether models truly extract input-independent computational principles that can be applied to novel contexts.

Chapter 4

How Models Learn to Reason from Code Data

4.1 Overview

In the previous chapter, we discover that LLM reasoning is unlike retrieval from parametric knowledge and relies on procedural knowledge extracted from pre-training data. In this chapter, we go one step further and investigate at what level of abstraction models can acquire knowledge from training data. Specifically, we ask whether models learn to execute procedures on novel inputs from exposure to abstract procedural representations alone, without ever seeing input-output examples for those specific procedures. This would clearly demonstrate generalisation from explicit procedural knowledge encountered in the training data — i.e. abstract, input-independent instructions such as code and mathematical formulas — to implicit use of this procedural knowledge at inference time, when these procedures are applied in specific contexts.

Code provides an ideal domain for studying this type of generalisation, as it naturally distinguishes between explicit and implicit procedural knowledge: source code represents input-general procedures, while program evaluations (i.e. execution traces) reveal how these procedures unfold step-by-step for specific inputs. Moreover, exposure to code has been increasingly recognised as a key driver of LLMs' reasoning abilities [Ary+24; PPF25]. Yet, a fundamental question remains unresolved: does code allow LLMs to internalise algorithmic abstractions that can be reused for reasoning across tasks, or does it simply offer a more concentrated form of logical reasoning compared to other forms of data?

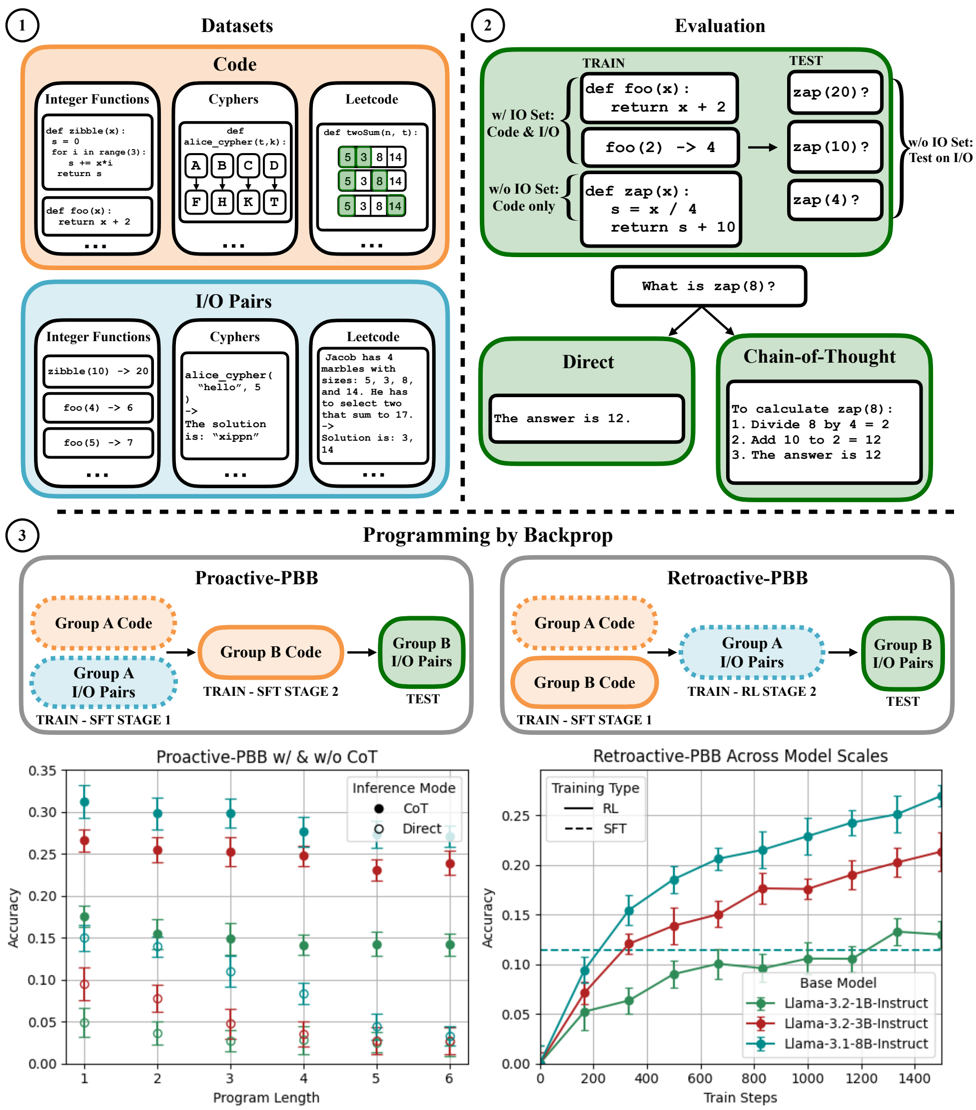


Figure 4.1: When LLMs are fine-tuned to auto-regressively predict source tokens for previously unseen programs, the ability to evaluate these programs for inputs also emerges, provided they are separately trained on I/O pairs for different programs. We call this *Programming by Backprop* (PBB), and demonstrate it on three datasets: random arithmetic programs, Leetcode programs, and custom ciphers. Models can learn to evaluate programs implicitly, executing multiple lines of code in the forward pass, as well as by using chain-of-thought reasoning. Generalisation happens to later seen code (Proactive-PBB) as well as earlier seen code (Retroactive-PBB), but the latter mainly if reinforcement learning is used. In this Figure, we use 'Group A' to refer to *w/ IO* programs (trained on as both code and I/O pairs), and 'Group B' to refer to *w/o IO* programs (trained on only as code, before testing on I/O pairs).

Our findings, summarised in Figure 4.1, demonstrate that training LLMs to evaluate programs, either *a-priori* through an SFT stage with programs and evaluations, or *a-posteriori* through RL on program evaluations, allows LLMs to evaluate programs they have only seen presented as source code, even when they are from a different domain. These findings reveal a generalisation from abstract source code to concrete applications, indicating a more sophisticated form of computational abstraction than previously documented.

4.2 LLMs as Interpreters: Programming LLMs by Training on Code

We study the extent to which LLMs behave as interpreters, evaluating programs that appear in their training data without I/O examples. To do so, the LLM must learn to perform input-specific execution of input-general procedures that are learned via next-token prediction — a process we call Programming by Backprop (PBB). Successful PBB implies that an LLM has not simply memorised the program's source code, but learned generalisable representations of the procedure it encodes at different levels of abstraction. We investigate PBB in the context of controlled fine-tuning experiments on pre-trained LLMs.

Consider a dataset of programs $\mathcal{D}_{\text{code}} = (\mathcal{D}_{\text{code}}^{w/\ IO}, \mathcal{D}_{\text{code}}^{w/o\ IO})$. We also define a dataset of I/O pairs $\mathcal{D}_{\text{exec}} = (\mathcal{D}_{\text{exec}}^{\text{train}}, \mathcal{D}_{\text{exec}}^{\text{test}})$ associated with the functions in $\mathcal{D}_{\text{code}}$. $\mathcal{D}_{\text{exec}}^{\text{test}}$ is the subset of I/O pairs we hold out and on which we test an LLM's ability to evaluate the programs in $\mathcal{D}_{\text{code}}^{w/o\ IO}$. $\mathcal{D}_{\text{code}}^{w/\ IO}$ and $\mathcal{D}_{\text{exec}}^{\text{train}}$ therefore facilitate learning a correspondence between program source code and program evaluations that may transfer to other programs the LLM has previously been or will later be trained on. Depending on whether or not chain-of-thought is used for inference, $\mathcal{D}_{\text{exec}}^{\text{train}}$ can feature outputs that include a chain-of-thought reflecting the entire program evaluation or simply the final output.

With this setup, we propose two approaches to eliciting PBB. The first, called Proactive-PBB, is a two-stage SFT pipeline (Algorithm 1). The first stage uses a mixture of programs and corresponding I/O pairs, priming the model to implicitly learn to evaluate programs seen in future training data. In the second stage, the resulting model is trained on other programs as source code alone.

The second approach, which we call Retroactive-PBB, is summarised in Algorithm 2. It consists of a first stage in which the model is trained via SFT on program source code alone, followed by a second stage of RL for program

Algorithm 1 Proactive-PBB

$\mathcal{D}_{\text{code}} = (\mathcal{D}^{w/\ IO}_{\text{code}}, \mathcal{D}^{w/o\ IO}_{\text{code}})$:= Dataset of program source code
$\mathcal{D}_{\text{exec}} = (\mathcal{D}^{\text{train}}_{\text{exec}}, \mathcal{D}^{\text{test}}_{\text{exec}})$:= Dataset of I/O pairs
$\mathcal{M}_{\text{base}}$:= pre-trained LLM
$\mathcal{M}_{\text{stage-1}} = \text{SFT}(\mathcal{M}_{\text{base}}, (\mathcal{D}^{w/\ IO}_{\text{code}}, \mathcal{D}^{\text{train}}_{\text{exec}}))$ // Proactively learn code-I/O relationship
$\mathcal{M}_{\text{stage-2}} = \text{SFT}(\mathcal{M}_{\text{stage-1}}, \mathcal{D}^{w/o\ IO}_{\text{code}})$ // Implicitly learn I/O mapping for new code
Metric = $\text{Accuracy}(\mathcal{M}_{\text{stage-2}}, \mathcal{D}^{\text{test}}_{\text{exec}})$

evaluation on the programs from $\mathcal{D}^{w/\ IO}_{\text{code}}$ via chain-of-thought[1]. This second stage enables the model to retroactively learn a general correspondence between learned code and input-specific problems. As demonstrated in Section 4.4, we find that RL is able to elicit this retroactive generalisation much better than SFT.

Algorithm 2 Retroactive-PBB

$\mathcal{M}_{\text{stage-1}} = \text{SFT}(\mathcal{M}_{\text{base}}, \mathcal{D}_{\text{code}})$ // Learn new code
$\mathcal{M}_{\text{stage-2}} = \text{RL}(\mathcal{M}_{\text{stage-1}}, \mathcal{D}^{\text{train}}_{\text{exec}})$ // Retroactively learn code-I/O relationship
Metric = $\text{Accuracy}(\mathcal{M}_{\text{stage-2}}, \mathcal{D}^{\text{test}}_{\text{exec}})$

4.3 Experimental Setup

4.3.1 Datasets

To empirically investigate PBB with LLMs, we generate several synthetic datasets ranging from arbitrary Python programs to real-world algorithmic reasoning problems. We create the following three datasets, each of which we open-source to facilitate future research:

- **Random Arithmetic**: randomly generated programs of varying length involving standard arithmetic and control flow.

[1] We do not test Retroactive-PBB without chain-of-thought inference, as we use RL fine-tuning.

- **Leetcode**: common algorithmic challenges found on competitive programming platforms, taken from an open-source dataset of Leetcode problems [HFg].

- **Ciphers**: custom encryption algorithms.

Each dataset includes programs as source code with a series of associated word problems representing the input/output pairs that assess the ability to evaluate the relevant program. Our focus is on generalisation from code, learned via SFT, to the word problems. At test time, the model needs to parametrically 'retrieve' the program learned via SFT and evaluate it step-by-step for inputs. Across all datasets, each word problem features a direct and chain-of-thought solution. Chain-of-thought reasoning is generated in a post-rationalised manner, using `GPT-4o` conditioned on the respective algorithm and the answer itself. All training datapoints are formatted as prompt-response pairs, meaning that program source code appears as a response to a prompt requesting an implementation of the program. Detailed descriptions of each dataset follow.

Random Arithmetic. This dataset includes 1000 unique Python programs synthetically generated to map integer inputs to integer outputs. Programs are constructed by composing fundamental Python control flow structures (e.g. `for` loops, `if` / `else` conditionals) with a set of standard algebraic operators (addition `+`, subtraction `-`, multiplication `*`, integer division `//`, modulo `%`, comparison `>`, `<`, exponentiation `exp`, and absolute value `abs`). Each program is categorised by its length, measured as the maximum number of operations executed for any input within the specified range. The dataset maintains a uniform distribution of programs across lengths. For each program, we provide 100 I/O word problems, 10 of which are reserved for test, with inputs spanning the integer range $[-50, 50)$.

Leetcode. This dataset consists of 702 Leetcode problems and their Python solutions. Problems range from easy to hard and have been filtered to ensure that each solution is a single function. For each Leetcode problem, we use `GPT-4o` to generate 20 word problems, 10 of which are reserved for test. The ground truth final solutions to these word problems are procedurally generated by running the relevant source code on the correct inputs.

Ciphers. To construct this dataset, we first take three common ciphers: Caesar, Atbash, and Vigenère. We then augment each real cipher, forming three custom ciphers:

- Alice: Similar to a Caesar cipher, but the index of each letter in the plaintext is added to the shift for that letter.
- Bob: Similar to an Atbash cipher, but each letter in the plaintext is replaced by its opposite letter in a *shifted* alphabet.
- Kevin: Similar to a Vigenère cipher, but the shift associated with a letter in the key is added to the corresponding plaintext letter if that shift is even and subtracted if it is odd.

Each cipher is parameterised by an integer shift and the Kevin cipher additionally takes a text key parameter. These custom ciphers allow us to perform controlled evaluation of an LLM's ability to do encryption for a cipher that we can reasonably assume is not within its pre-training data. For each cipher, we generate 400 encryption problems, 300 of which are reserved for test. The test split for this dataset is much larger because there are far fewer programs.

4.3.2 Training Details

Below, we provide the main training details for experiments on each dataset. Further implementation details and hyperparameters are included in Appendix C.1. For each experiment, we perform evaluation across 16 generations per problem, sampled with temperature 0.8, and we report 95% confidence intervals over these. Due to computational constraints, all training is done for a single seed.

Random Arithmetic. For our main experiment, each group of programs (*w/ IO* and *w/o IO*) consists of 100 functions, but we also explore the impact of data scaling in Appendix C.2. The 10 test word problems for each *w/o IO* program are used for final evaluation. We use the 1B, 3B, and 8B instruction-tuned models from the `Llama 3` series [Dub+24] as the base models.

For Proactive-PBB, each of the 90 train word problems for *w/ IO* programs are used in stage 1 training. Each function definition appears in the training data 30 times, with different augmentations being applied to the prompt (e.g. "Provide a Python function that applies <`program name`> to a number.") and response preamble (e.g. "Here is a function that accomplishes that: `<source code>`"). These augmentations ensure that the same code appears as a number of distinct datapoints, which we find is crucial for eliciting program distillation (Figure 4.2 left). To preserve instruction-following capabilities during stage 2 (code-only fine-tuning) we mixed in 1000 MathInstruct samples [Tos+24], which proves crucial for eliciting successful program evaluation. Each SFT stage involves a single epoch over the appropriate data.

For Retroactive-PBB, given that stage 2 is done with RL, we only consider chain-of-thought program evaluation. For the RL stage, we use GRPO [Sha+24] with a group size of 6 and batch size of 6, meaning that each batch consists of a single group. We therefore use only 15 I/O pairs (90/6) per *w/ IO* program for RL training. This is so that we can make a fair comparison to instead using SFT for stage 2, where we only train on a single chain-of-thought per input.

Leetcode. For Leetcode experiments, 500 randomly selected programs are used as the *w/ IO* group and 100 as the *w/o IO* group. Other aspects of the training setup are the same as above.

Ciphers. We use the ciphers dataset to evaluate transfer of the ability to evaluate programs for inputs across different algorithmic domains. We perform stage 1 of Proactive-PBB on Leetcode data (*w/ IO*) and then perform stage 2 on the cipher source code (*w/ IO*).

Our use of ciphers is motivated by McCoy et al. [McC+24], who show that LLMs, having learned to use ciphers to encrypt text from naturally occurring examples in pre-training data, are biased by the uneven distribution of algorithmic parameters in these examples — a phenomenon termed the 'embers of autoregression'. Given that program distillation opens the door to learning programs such as ciphers from their source code alone, we investigate whether it can overcome these embers. We therefore also create a dataset of encryption problems and correct chain-of-thought responses, where the shift parameter used in each cipher is drawn from a Gaussian distribution with mean 13 and standard deviation 2 (mirroring the fact that `ROT13` is a far more frequently occurring shift in real-world data). The frequencies of sampled shifts are reported in Appendix C.5. We then perform SFT on this biased dataset of examples and compare the test execution accuracy to that obtained by Proactive-PBB (stage

1 on Leetcode, stage 2 on Ciphers). For this experiment, we fine-tune `GPT-4o` via the OpenAI fine-tuning API, as we find that `Llama-3.1-8B-Instruct` is unable to perform accurate encryption for the ciphers we consider, even when trained on many examples.

4.4 Results

In this section, we discuss the results of our experiments. The main overarching finding is that generalisation from autoregressive next-token prediction on abstract source code to concrete applications of these programs for input-output pairs happens, provided models are separately trained on concrete applications of a different set of program source codes (which may be of a different domain). We start by discussing the results on the random arithmetic dataset, followed by experiments testing generalisation from evaluating Leetcode programs to cipher algorithms.

4.4.1 Random Arithmetic Results

Finding 1: Generalisation from abstract source code to concrete applications improves with model scale, and drops off with program length.

We first evaluate how well models of different scales use Proactive-PBB to learn to evaluate the random arithmetic programs for which no word problems (and hence, no input-output pairs) have appeared in training. These results are in Figure 4.1 (bottom left). We observe that the ability to solve problems

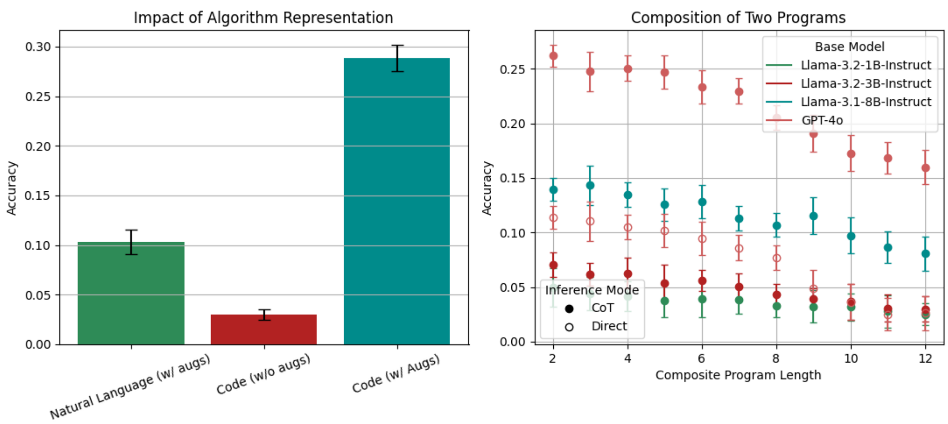


Figure 4.2: Left: Accuracy of `Llama-3.1-8B-Instruct` following Proactive-PBB on evaluating *w/o IO* random arithmetic programs represented as natural language or code. The use of data augmentation is also ablated. **Right**: Accuracy of different models following Proactive-PBB for compositions of two programs that have been trained on independently.

by evaluating these programs improves with model scale. This is the case both when the model can do so in-context via chain-of-thought and when the model has to directly output the answer, implicitly evaluating the program in the forward pass. For both forms of inference, shorter programs are easier to program into the model. At the 1B model scale, there is little to no capacity for implicit program evaluation. For the 3B and 8B models, the accuracy–program length curves are flatter for chain-of-thought evaluation, indicating that explicit reasoning increases the capacity for these models to handle successive computational operations. In Appendix C.3, we show that the two-stage approach of Proactive-PBB means that the same piece of code from $\mathcal{D}_{\text{code}}^{w/o\ IO}$ can appear fewer times in the training data than if the data is mixed in a single fine-tuning stage. This suggests that a data curriculum encouraging a general code-I/O relationship to be learned proactively enables more efficient learning of program evaluation for programs seen only as source code.

Finding 2: Generalisation from abstract programs to concrete applications works better if programs are represented in code rather than equivalent natural language descriptions.
We next investigate the impact of how programs are represented in training data on the effectiveness of Proactive-PBB. We do so by training on semantically equivalent natural language program descriptions instead of code functions (example included in Appendix C.6). In Figure 4.2 (left), we see that `Llama-3.1-8B-Instruct` is much better at being programmed with code than with natural language descriptions, despite the fact that the programs are also perfectly described in natural language. This could indicate that the structure and syntax of code enables LLMs to more easily internalise the algorithmic abstractions it represents. However, given that we only investigate PBB at the fine-tuning stage, it is unclear to what extent this result is due to an emphasis on code training in the construction of the base model.

Finding 3: Presenting the same piece of code with multiple prompt augmentations is crucial for the generalisation from source code to applications to happen.
We also ablate the use of prompt and response preamble augmentations, shown in Figure 4.2 (left). We find that the augmentations used to create multiple distinct datapoints with the same program source code is essential.

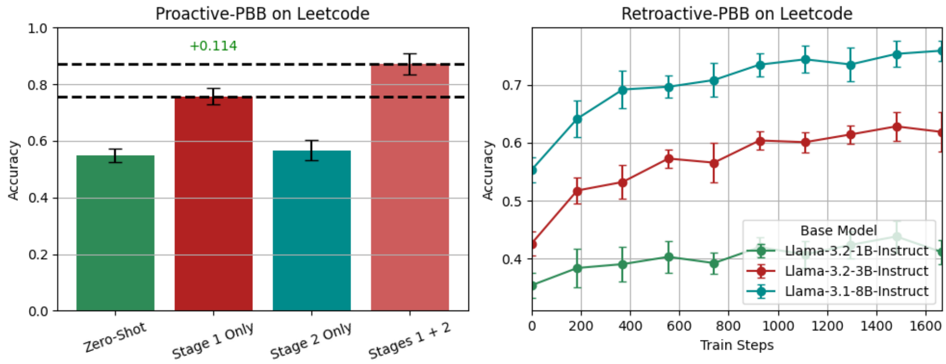


Figure 4.3: **Left**: Accuracy of `Llama-3.1-8B-Instruct` following each stage of Proactive-PBB on evaluating *w/o IO* Leetcode programs. **Right**: Accuracy of different models at evaluating *w/o IO* Leetcode programs following Retroactive-PBB.

Berglund et al. [Ber+23] similarly find data augmentations to be useful for out-of-context-reasoning (of which the generalisation discovered here is an example, discussed below in Section 4.5), however our results show additionally that it is not necessary to augment the source code itself: simply augmenting the prompt and the response text preceding the source code is enough.

Finding 4: Generalisation from training on source code to applications of compositions of these source codes also happens. For the Llama models, this only works if they are allowed to use chain-of-thought at inference time, but `GPT-4o` can sometimes evaluate compositions of programs entirely within its weights.

We also study the effectiveness of Proactive-PBB on composite functions. Here, the set of programs seen only as code includes functions defined as the composition of two of the original random arithmetic functions. E.g. the composition of zibble and foo from Figure 4.1 would appear as follows:

```
def bar(x):
  return zibble(foo(x))
```

For a given task, the model must therefore parametrically retrieve the composite function definition, as well as both of the functions being composed, so that it can execute one and use the resulting input for the other. We find that the `Llama` models we consider fail to evaluate composite programs implicitly, but they are sometimes capable of doing so when using chain-of-thought (Figure 4.2 right). We check whether much larger models are capable of evaluating composite programs implicitly by following the same fine-tuning steps with `GPT-4o`. Remarkably, `GPT-4o` demonstrates some ability to successfully

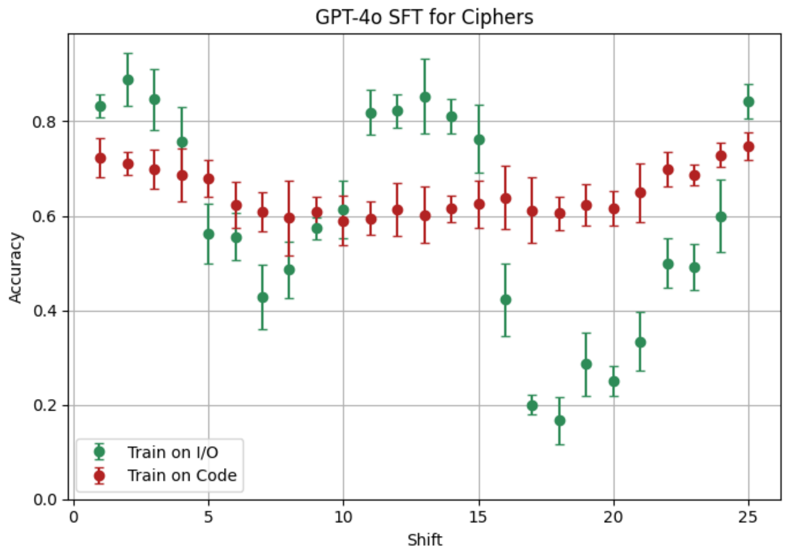


Figure 4.4: Accuracy of `GPT-4o` when encrypting text with ciphers trained on only as code, or when trained on as demonstrated execution traces with unevenly distributed shifts.

retrieve two programs that were trained on independently and successively evaluate each, all without producing an explicit chain-of-thought.[2] As expected, it can do composite program evaluation much better when it generates the intermediate reasoning steps. In all cases, we see that composition is harder when the composite program length is greater, which we attribute to the likelihood of errors growing as the number of successive operations increases.

Finding 5: Generalisation to applications of earlier-seen program source codes happens primarily if the second stage of training on programs with I/O pairs is done with reinforcement learning.
Finally, we test how well models of different scales use Retroactive-PBB to learn to evaluate random arithmetic programs previously trained on (Figure 4.1 bottom right). Performance again varies considerably with model scale. We see that the final performance of the 8B model after fine-tuning with SFT in the second stage is surpassed even by the 1B model when RL is used instead. This is a strong indication that the use of on-policy data in RL is supporting generalisation, which aligns with hypotheses and findings in prior work [HH63a; OCD21; Kir+24; Chu+25]. We investigate this further in Appendix C.4 by running DPO to isolate the roles of learning from negative samples and

[2] As `GPT-4o` is an API-based model, we cannot be certain that it is not using additional inference compute.

learning from on-policy data: whilst DPO considerably outperforms SFT, GRPO remains the most effective algorithm, indicating that both negative samples and being on-policy are beneficial.

4.4.2 Leetcode and Cipher Results

Finding 6: When fine-tuned on Leetcode programs and their I/O pairs, `Llama-3.1-8B-Instruct` improves in applications of other Leetcode programs that must have been seen during pre-training. In Figure 4.3 (left), we show how the outcome of Proactive-PBB on Leetcode programs with the 8B model varies with each stage. Note that this is the only dataset for which all programs likely appear in the pre-training data, and I/O pairs for each may appear, even the *w/o IO* set of programs. The Leetcode programs are mainly used to investigate generalisation to another set of *w/o IO* programs, discussed below. However, we discuss the results on Leetcode alone first. Stage 1 alone (i.e. *w/ IO* code and I/O pairs) yields a significant performance jump when evaluating the *w/o IO* programs. Given all of the Leetcode programs in our dataset are probably within the model's pre-training data, some generalisation from stage 1 alone is expected. This result shows that generalisation from stage 1 to source code seen during pre-training happens, although it is unclear whether input-output pairs have been seen for these. We do however observe a further performance increase following stage 2 (*w/o IO* code), where the program source code associated with the held-out word problems is trained on.

In Figure 4.3 (right), we show how test accuracy increases during Retroactive-PBB for different model scales. Given that zero-shot accuracy is non-zero for Leetcode word problems, each model also starts at a different initial performance following stage 1 SFT. However, we observe that the progress made by the 3B and 8B model over the course of stage 2 RL is much greater than that of the 1B model, indicating that its more limited capacity restricts its ability to generalise to the programs that are only trained on as source code.

Finding 7 & 8: Generalisation from source code to applications emerges even if the training stage with I/O pairs enabling this generalisation uses programs from another domain. Moreover, teaching models algorithms in this way leads to a more uniform performance

across inputs than training on I/O pairs for the programs. We report the results for transfer of Leetcode program evaluation abilities to novel ciphers via Proactive-PBB with `GPT-4o` in Figure 4.4 ('Train on Code'). As the ciphers are custom, with made-up names, zero-shot encryption accuracy prior to fine-tuning is zero. Firstly, we note that, following stage 1 training on Leetcode programs with I/O pairs, training on the custom ciphers source code alone is sufficient for the model to encrypt text at reasonable accuracy. Secondly, when comparing to the performance following training on the dataset of chain-of-thought solutions with an uneven distribution of shifts ('Train on I/O'), we see that PBB yields a much more uniform accuracy across shifts. This demonstrates that PBB is a promising direction for addressing the 'embers of autoregression'. However, the peak performance of training on I/O is higher, indicating that demonstration data is advantageous when there are sufficient examples for each parameter variation. Whether training on code or I/O, we see that performance increases for the smallest and largest shifts, because these shifts point to letters that are close to the original in the alphabet.

4.5 Related Work

Training LLMs on code. Code is a standard part of most pre-training corpora [Dub+24; Gem+25; Coh+25; Qwe+25, inter alia] and believed to be an important contribution to general-purpose reasoning abilities. Although this is mostly conventional wisdom, some prior work has shown that training on code positively transfers to downstream tasks [Ary+24]. Further, Petty, Steenkiste, and Linzen [PSL24] hypothesise that compositionality in code helps with learning to generate structured outputs and mathematical reasoning, presenting experimental results supporting this. In this work, we hypothesise code has the complimentary advantage of explicitly representing procedures that can be reused (i.e. implicitly executed) for step-by-step reasoning problems.

Out-of-context reasoning. Our work has deep connections to the growing body of work that investigates out-of-context reasoning (OOCR) in LLMs [AL23; Ber+23; Tre+24; Bet+25b; Bet+25a]. This can be loosely defined as an LLM's ability to infer knowledge implicit within its training data and apply that knowledge downstream, without in-context demonstrations. We consider program source code and show that training on this allows models to encode knowledge of how to compute outputs for specific inputs. A second way in which our setting evaluates OOCR is the ability of models to accurately generate outputs for a program seen in its training data by executing it implicitly for specific inputs entirely within its weights (no chain-of-thought reasoning). Prior to LLMs being developed, Zaremba and Sutskever [ZS15] study the ability of LSTMs trained from scratch to execute programs adding two 9-digit numbers in one forward pass through the model. Wang et al. [Wan+24b] study similar implicit reasoning behaviours in grokked transformers, where the training data containes I/O samples rather than explicit procedures. Krasheninnikov et al. [Kra+24] investigate a phenomenon similar to OOCR that occurs during the training process itself, which they refer to as implicit meta-learning. Similarly to our work, they use a two-stage fine-tuning process to first train an LLM to learn associations between aspects of the data (in their case, tokens implicitly indicating truthfulness, in ours, function definitions and calls), and then further train the LLM such that its learning will be informed by those associations. Lampinen et al. [Lam+25] investigate distilling in-context reasoning abilities into OOCR abilities. Our work similarly considers problems where LLMs generalise in-context information more effectively than information that has only appeared in their training data.

Emergence of algorithmic reasoning. Previous research into the emergence of algorithmic reasoning during training has revealed a number insights. By training on search traces, Gandhi et al. [Gan+24] show that language models can learn to emulate the search process, thus demonstrating algorithm distillation [Las+23] via next-token prediction. By training transformers on variable dereferencing problems, Wu, Geiger, and Millière [WGM25] illuminate three distinct phases of learning, as models transition from using heuristics to implementing systematic variable binding. Meanwhile, some prior work illuminates critical failures in the algorithmic reasoning of language models. Through mechanistic interpretability experiments, Nikankin et al. [Nik+25] find that language models may use heuristics to solve arithmetic problems, rather than doing so algorithmically. Shojaee et al. [Sho+25] similarly show that LLMs fail to use explicit algorithms and reason inconsistently. Perhaps relatedly, McCoy et al. [McC+24] reveal that LLMs suffer from the 'embers of autoregression', causing their outputs on algorithmic tasks to depend on the probability of the inputs, algorithms, outputs under the pre-training data distribution. Finally, Thomm et al. [Tho+24] show that language models can fail to compose algorithms they have learned independently. This is in contrast to our finding that LLMs can in fact do so for algorithms independently learned through backpropagation (Figure 4.2, right).

4.6 Discussion, Limitations, and Future Work

This chapter demonstrates that LLMs develop reusable algorithmic abstractions from code training, providing an explanation for how training on code may lead to improved general-purpose reasoning. Importantly, we do not argue our fine-tuning experiments themselves confer novel reasoning abilities; rather, we demonstrate how fine-tuning on code might provide models with abstractions they can leverage at inference time — even without accompanying I/O demonstrations. Several limitations warrant discussion.

Synthetic datasets. We deliberately construct datasets featuring novel algorithms, composed of random or synthetically chosen sequences of operations, to minimise overlap with the model's pre-training corpus. This controlled approach provides a compelling story for how models might improve their reasoning abilities, since real-world algorithms can similarly be broken down into sequences of basic operations like the ones in our datasets. Whilst our experiments on Leetcode problems and ciphers take steps towards real-world domains, the application of this approach to more complex practical algorithms presents a promising avenue for future research.

Limited performance on program evaluation. Although we convincingly demonstrate even small models can be programmed by backpropagation, the performance is still limited. Future work can investigate how we can obtain models that are more amenable to being programmed, perhaps by incorporating procedurally generated source code — I/O data mixtures into pre-training.

Fine-tuning vs pre-training. Our experiments are conducted in a controlled fine-tuning regime. The inductive biases of this setup differ from those of large-scale pre-training, and it is not clear to what degree models internalise reusable algorithmic abstractions during pre-training. Nonetheless, we believe similar mechanisms may be at play during pre-training, as the data is likely to contain many copies of common algorithms and corresponding execution traces for at least some of them. Further, our results showing transfer from one domain of programs to another make Programming by Backprop from natural pre-training data more likely.

Future work. Our work leaves open many questions about whether and how transfer from program generation to program evaluation manifests during pre-training, as well as potential ways to exploit the idea for model training. For example, do models indeed internalise reusable and relevant abstractions from code during pre-training, like those encoded by search or planning algorithms? Further, can we generate synthetic program source code that aids internalisation of useful abstractions? If so, a single LLM could in principle be used to generate and internalise novel programs in a loop similar to those that maintain an external database of programs [Nov+25]. Finally, we find that Programming by Backprop is much more effective from code data than semantically equivalent programs described in natural language. This spurs interesting questions about what makes code special, and whether this insight can be further exploited besides reasoning problems. For instance, can we align models to constitutional principles if these are presented symbolically?

This and the previous chapter together demonstrate that LLMs can acquire generalisable mathematical and algorithmic principles from pre- and post-training and apply these in novel contexts. Chapter 3 demonstrates that even if the answers to reasoning steps appear in the pre-training data, it does not mean the model is relying on them to produce reasoning chains. In fact, their generalisation strategy looks entirely different than factual retrieval. This chapter further confirms that models can generalise from program generation to program evaluation, demonstrating that the procedural knowledge models acquire during auto-regressive training abstracts away from specific inputs. However, the types of reasoning studied in these chapters are both formal, underlying strict axioms and logic. The next chapters investigate a different type of reasoning where objective standards of correctness are not well-defined.

Chapter 5

A Case Study in Social Reasoning: Pragmatics

5.1 Overview

The reasoning tasks examined in the previous two chapters feature objectively correct steps and answers, providing an ideal testing ground for determining whether models rely on training data exposure to the answers and for understanding the levels of abstraction across which models can generalise. However, a more complete characterisation of LLM reasoning requires investigating tasks that humans perform naturally but for which correctness depends on social consensus rather than logical necessity. To this end we turn to social reasoning in the next two chapters, starting with pragmatic language understanding.

Consider the following exchange:

Esther: "Can you come to my party on Friday?"
Juan: "I have to work."

Meaning in language is not only determined by a combination of words, but also context, beliefs, and social institutions [Wit53; Gri75; Hua17]. We resolve Juan's response as him declining the invitation by using the contextual commonsense knowledge that having to work on a Friday night precludes attendance. This exchange contains an *implicature* — utterances that convey something other than their literal meaning.[1] Implicatures illustrate how context contributes to meaning; distinguishing writing and speaking from communicating [Gre96]. We cannot fully understand utterances without understanding their implications. Being able to resolve completely novel implicatures and,

[1]In Appendix D.1 we present an introduction to implicature.

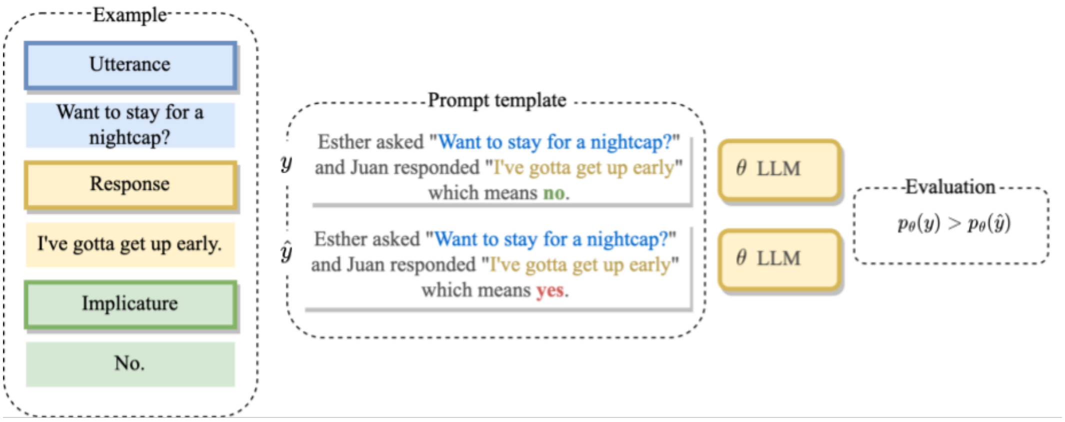


Figure 5.1: A schematic depiction of the protocol we propose to evaluate whether language models can resolve implicatures. Each example in the test set gets wrapped in templates and transformed into an *incoherent* example by swapping "yes" and "no". The model is said to resolve the implicature if it assigns a higher likelihood to the coherent text than the incoherent text.

more broadly, engage in pragmatic understanding constitutes an essential and ubiquitous aspect of our every day use of language.

This raises an important question: *to what extent can large language models resolve conversational implicature?* To answer this question we use a public dataset of conversational implicatures and propose an evaluation protocol on top of it (Figure 5.1). We evaluate a range of models that can be categorised into four groups; large-scale pre-trained models, like OPT [Zha+22], LLMs fine-tuned on conversational data, like BlenderBot [Ng+19], LLMs fine-tuned on common NLP benchmarks with natural instructions for each benchmark, like Flan-T5 [Chu+22], and LLMs fine-tuned on tasks with natural instructions for each example, e.g. versions of OpenAI's InstructGPT-3 series[2]. Our results show that implicature resolution is a challenging task for LLMs. All base models obtain close to random zero-shot accuracy (around 60%), whereas humans obtain 86%. Base models' performance improves with model scale and with few-shot in-context examples. However, our results suggest that instruction-tuning at the example level is important for pragmatic understanding. Models fine-tuned with this method perform much better than others, and analysis of different model sizes shows that they have the best scaling properties. We further push performance for these models with chain-of-thought prompting,

[2]The precise method is unpublished and differs from the original instructGPT [Ouy+22].

and find that one model in the group (GPT-4) reaches human-level performance. In summary, we conclude that human-like pragmatic understanding has not yet arisen from large-scale pre-training *on its own*, but scaling analysis shows that it might for much larger scale. Fine-tuning on conversational data or benchmark-level instructions does not produce models with pragmatic understanding. However, fine-tuning on instructions at the example-level is a fruitful path towards more useful models of human discourse.

5.2 Evaluation Protocol

Here we outline the evaluation protocol we use to answer the question "To what extent can LLMs resolve conversational implicature?". We focus on binary implicatures that imply "yes" or "no" (see Figure 5.1). We say a model resolves an implicature correctly if it assigns higher likelihood to a coherent utterance than a similar but incoherent one, detailed below.

Zero-shot evaluation. Consider the example from the introduction packed into a single utterance:

> Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means no.

We can transform this example to be *pragmatically incoherent* (in the sense that it will become pragmatically inconsistent with expected use) by replacing the word "no" with "yes":

> Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means yes.

To resolve the implicature, the model should assign higher likelihood to the first of the two sentences above, namely the most coherent one. Importantly, both sentences have exactly the same words except for the binary implicature "yes" or "no", making the assigned likelihood scores directly comparable. Formally, let the coherent prompt be y and the augmented, incoherent prompt be $\hat{y}$. A model outputs a likelihood p parameterised by weights θ. We say a model correctly resolves an example y when it assigns $p_\theta(y) > p_\theta(\hat{y})$. This is equivalent to evaluating whether the model assigns a higher likelihood to the correct continuation of the two options. Note that this is a more lenient evaluation protocol than sometimes used for language models, where models are evaluated on on their ability to generate the correct continuation, in this case "no". The greedy decoding approach (evaluating whether "yes" or "no" is generated) is also captured by our approach, but we additionally label an example correct if "no" is not the highest assigned likelihood, but still higher than "yes". We did not opt for greedy decoding because "no" is not the only coherent continuation here, and marginalising over all possible correct continuations is intractable. The more lenient evaluation does capture implicature resolution, because the choice of "no" versus "yes" is only determined by the resolution of the implicature. We guide the models to output "yes" or "no" explicitly in three of the six prompt templates with instructions, such that we can estimate the effect of this guidance on performance. For two model classes (i.e. GPT-3.5-turbo and GPT-4) we do not have access to likelihoods, and for these models we take the greedy decoding approach, guiding the model to output "yes" or "no" explicitly in all prompts (see Table D.2 in Appendix D.2).

We use a dataset of conversational implicatures curated by George and Mamidi [GM20].[3] It contains implicatures that, like in Figure 5.1, are presented in utterance-response-implicature tuples. Of these, 718 are binary implicatures that we can convert into an incoherent sentence. We randomly sample 600 examples for the test set and keep the remaining 118 as a development set to improve implicature resolution after pre-training through in-context prompting or fine-tuning.

Few-shot in-context evaluation. We add k examples of the task to the prompt, e.g. with $k = 2$:

> Esther asked "Have you found him yet?" and Juan responded "They're still looking", which means no.
>
> Esther asked "Are you having fun?" and Juan responded "Is the pope Catholic?", which means yes.
>
> Finish the following sentence:
>
> Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means no.

We evaluate the models' k-shot capabilities for $k \in \{1, 5, 10, 15, 30\}$ by randomly sampling k examples from the development set for each test example. We opt for a random sampling approach to control for two sources of randomness.

[3]Published under a CC BY 4.0 license.

Firstly, examples have different levels of informativeness. Secondly, recent work found that the order in which examples are presented matters [Lu+22]. Ideally, to marginalise over these random factors, we would evaluate each test example with all permutations of k examples from the development set. This requires $\frac{118!}{(118-k)!}$ evaluations for each test example, which is intractable. Instead, we estimate performance per test example by randomly sampling from the development set. In this way we control for some of the variance in performance, but avoid extra evaluations. We ensure each model sees the same few-shot examples per test example.

Controlling for prompt sensitivity. It has been shown language models are sensitive to prompt wording [EL20; Tan+21; RM21a; WP21]. To control for this factor of randomness we manually curate six different template prompts and measure performance across these. One of the templates has been presented above, namely "Esther asked <*utterance*> and Juan responded <*response*>, which means <*implicature*>". Another template is: "Question: <*utterance*>, response: <*response*>, meaning: <*implicature*>". The former we call *natural* prompts and the latter *structured* prompts. Each group has three templates that only differ slightly in wording. This grouping allows us to look at the variance due to slight changes in wording as well as performance difference due to a completely different way of presenting the example. The full list of prompts can be found in Appendix D.2.

5.3 Experiments

The set of large language model classes we evaluate can be grouped into four distinct categories:

1. *Base models*: large-scale pre-trained models; RoBERTa [Liu+19], BERT [Dev+18], GPT-2 [Rad+19], EleutherAI [WK21; Bla+22], BLOOM [Big22], OPT [Zha+22], Cohere's base models, and GPT-3 [Bro+20a]

2. *Dialogue FT*: LLMs fine-tuned on dialogue, BlenderBot [Ng+19].

3. *Benchmark IT*: LLMs fine-tuned on tasks with natural instructions for each benchmark or "benchmark-level instruction-tuned models"; T0 [San+22] and Flan-T5 [Chu+22].

4. *Example IT*: LLMs fine-tuned on tasks with natural instructions for each example or "example-level instruction-tuned models"; a subset of

OpenAI's API models and Cohere's API models). A type of SFT where both the instructions and the completions are human-written.

Benchmark IT can be seen as a type of SFT where annotators write a single instruction for an entire dataset. The models are then fine-tuned on each example from the dataset with the same instruction. We distinguish this from example-level IT; the type of SFT where each example in the dataset has a human-written instruction and completion, resulting in a more diverse dataset. Each group contains model classes for which we evaluate a range of sizes. A detailed categorisation of the models and their attributes can be found in appendix D.3.[4] We make use of the OpenAI and Cohere APIs as well as the pre-trained models in the transformers library [Wol+20] and EleutherAI's framework to evaluate them [Gao+21]. All code used for this paper can be found on GitHub[5] and the dataset is made publicly available on HuggingFace[6]. Below, we present zero-shot and few-shot results, discussing patterns of performance of the models in the four different groups. We further look at the results for different model sizes of each model class and the variance over the prompt templates.

We contrast the models' performance with human performance. To this end, each test example gets annotated by five humans. We split the test set in four and assign each annotator a subset, leaving us with twenty annotators in total. The average human performance is 86.2%, and the best performance is 92%. Some of the errors humans make uncover examples that have multiple interpretations, and others uncover annotation errors. The nature of the task of implicature resolution means we do not expect models to perform better than human best performance. Details on the human experiment can be found in the Appendix D.4 (also containing an analysis of human errors), and detailed results per model and prompt template in Appendix D.7.10. We also test for spurious correlations present in the benchmark (like lexical cues the model can rely on, Appendix D.7.8). We do this by running the benchmark using utterance- or response-only examples, finding that models mostly perform random for the former, and are able to resolve response-only implicatures with about 65.5% accuracy (resolving implicatures with responses

[4]Note that there are several important aspects unknown for models behind APIs, like OpenAI's model sizes.

[5]https://github.com/LauraRuis/do-pigs-fly

[6]https://huggingface.co/datasets/UCL-DARK/ludwig

such as "do fish swim?"). The findings from this experiment additionally serve as a test for memorisation; if the models had memorised the benchmark they would likely perform above random in the utterance-only experiment.

Table 5.1: The k-shot accuracy ($k \in \{0, 1, 5\}$) for the best performing model of each class. For each model, we select the model size to show by choosing the one that achieves the best 5-shot performance. The std is over prompt templates for the models and over annotators for humans. FT stands for fine-tuning and IT for instruction-tuning. We find that the models in the *Example IT* class obtain significantly higher performance than all others. $\star$ means size unknown.

	Model	0-shot	1-shot	5-shot
Baseline	Random		50%	
Topline	Human avg.		86.2% ± 2.3	
Base models	BERT-110M	54.8% ± 1.6	51.7% ± 1.7	53.3% ± 2.2
	RoBERTa-355M	55.6% ± 2.0	54.1% ± 0.9	57.1% ± 1.5
	GPT-2-xl	51.3% ± 2.9	57.4% ± 3.3	57.7% ± 1.1
	EleutherAI-20B	57.5% ± 3.3	55.9% ± 2.3	61.1% ± 4.9
	BLOOM-176B	54.2% ± 1.2	61.1% ± 2.7	65.4% ± 3.4
	OPT-13B	61.0% ± 5.5	60.6% ± 2.7	67.4% ± 2.1
	Cohere-52B	58.5% ± 4.0	63.0% ± 3.8	65.1% ± 2.9
	GPT-3-175B	57.7% ± 4.4	65.7% ± 1.4	68.7% ± 1.5
Dialogue FT	BlenderBot-2.7B	53.4% ± 0.3	53.3% ± 0.1	53.3% ± 0.1
Benchmark IT	T0-11B	55.6% ± 7.0	47.8% ± 0.5	47.0% ± 0.2
	Flan-T5-11B	60.8% ± 2.4	57.4% ± 5.0	61.7% ± 4.8
Example IT	text-davinci-001-$\star$	72.3% ± 2.8	72.7% ± 1.3	74.5% ± 1.0
	text-davinci-002-$\star$	70.6% ± 2.3	75.6% ± 2.8	79.6% ± 2.0
	text-davinci-003-$\star$	71.2% ± 2.8	74.3% ± 1.4	79.7% ± 0.6
	ChatGPT-$\star$	72.1% ± 5.9	75.1% ± 1.5	73.9% ± 6.3
	GPT-4-$\star$	**81.8% ± 1.8**	**82.3% ± 1.4**	**82.0% ± 1.7**
	Cohere-command-52B	60.2% ± 5.2	72.8% ± 1.3	75.4% ± 1.8

Finding 1: Models instruction-tuned at the example level outperform all others.

Table 5.1 shows the best 0-, 1-, and 5-shot accuracy each model class achieved on the implicature task. The best overall accuracy is achieved by GPT-4 (the size of this model is unknown) at $82.3\% \pm 1.4$. This leaves a gap of 3.9% with human average performance. All models in the class *Example IT* perform significantly better than any of the other models for all k, except Cohere-command-52B at 0-shot. This result is more clearly seen in Figure 5.2, where we present the average accuracy for each model group. The performance for the other model classes across k ranges from 47.0% by BlenderBot-2.7b at $k = 5$ and 68.7% by GPT-3-175b at $k = 5$. Even though base models benefit from few-shot examples, their performance remains mostly closer to random than to humans for all k, showing a gap of at least 17.5% with the average human.

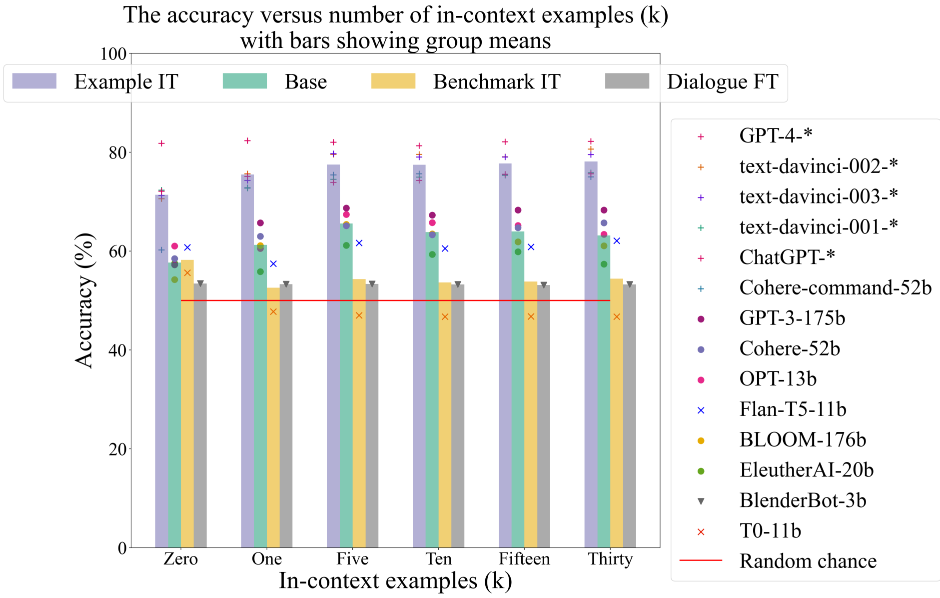


Figure 5.2: The few-shot accuracy for the best model of each class (e.g. the best performing model in the class Cohere-command is the 52B model, whereas the best model in the class OPT is the 13b model). The bars show the group means. Models fine-tuned on example-level instructions perform better than most other models, especially for $k > 0$. For all models there is a significant gap between best accuracy and human accuracy (which is 86.2%). * means size unknown.

We observe a decrease in performance for $k > 0$ for the group *Benchmark IT*. This is not surprising, as these kind of models are specifically fine-tuned to be better at zero-shot generalisation [San+22; Chu+22]. BlenderBot, in the group *Dialogue FT*, performs barely better than random for all k. We hypothesise that the lower performance which Cohere-command-52B achieves 0-shot is not due to a lack of implicature understanding, but due to a failure to calibrate the yes/no likelihoods without examples. For this model, we observe a sharp rise in performance from $k = 0$ to $k = 1$ (see Table 5.1 or Figure 5.2). Since it is unlikely that one example of an implicature induces pragmatic understanding, we hypothesise that few-shot prompting mostly serves to clarify the task format. We test this hypothesis in Appendix D.7.6 by repeating the 1- and 5-shot experiment with random labels for Cohere-command-52B and text-davinci-001. We find that the performance does not degrade, which confirms that the few-shot examples mainly serve to prime the model towards producing outputs following the yes/no structure.

Finding 2: The results are robust to different prompt templates. As detailed in Section 3, each example in the test set is wrapped in six different prompt templates. The standard deviation in Table 5.1 and in Figure 5.2 shows the sensitivity to different prompt wording. The standard deviation ranges from 0.3 for BlenderBot to 7.0 for T0-11B. All in all, the sensitivity to prompt wording does not seem to be a problem for this task; when taking into account the confidence intervals the result remains that models in the group *Example IT* perform significantly better than all other models, but worse than humans. In Appendix D.7.4 another analysis is presented that shows how different prompt templates benefit from in-context examples. The takeaway from the analysis is that few-shot prompting can mitigate the fact that some models are better at natural prompts and others better at structured prompts by improving performance on the type of prompt the model struggles with zero-shot. Again, when only looking at the best prompt type for each model class (i.e. structured or natural), the results remain that models in the group *Example IT* perform best.

Finding 3: Models instruction-tuned at the example-level have the most favourable scaling properties, but some base models also show positive correlation with scale. Figure 5.3 shows the scaling behaviour of the model classes for which we know the number of non-embedding parameters. We highlight 0- and 5-shot results, because for $k > 5$ the accuracy of most models plateaus (see Figure 5.2). However, detailed results for other k can be found in Appendix D.7.10. Note that we do not know the number of parameters for OpenAI's 'text-<engine>-001'-series, but we do know the order of the engines in size, and we separately present its scaling results in Table 5.2. Except OpenAI's 'text-<engine>-001'-series, none of the models show significant performance increase with model size for the 0-shot evaluations. However, for k-shot evaluations with $k \geq 1$ we observe significant positive correlation with size for the models in the *Example IT* class for which we have multiple sizes (Cohere-command and 'text-<engine>-001') as well as some models in the base model class. Not only do the models in the *Example IT* class exhibit higher performance for the same model size, these models also have a steeper performance increase with size than the base models. Comparing the scaling properties of the best base model (GPT-3) with Cohere-command, we see that the increase in performance from the second-largest to the largest model is 0.04% per billion parameters from GPT-3-6.7B

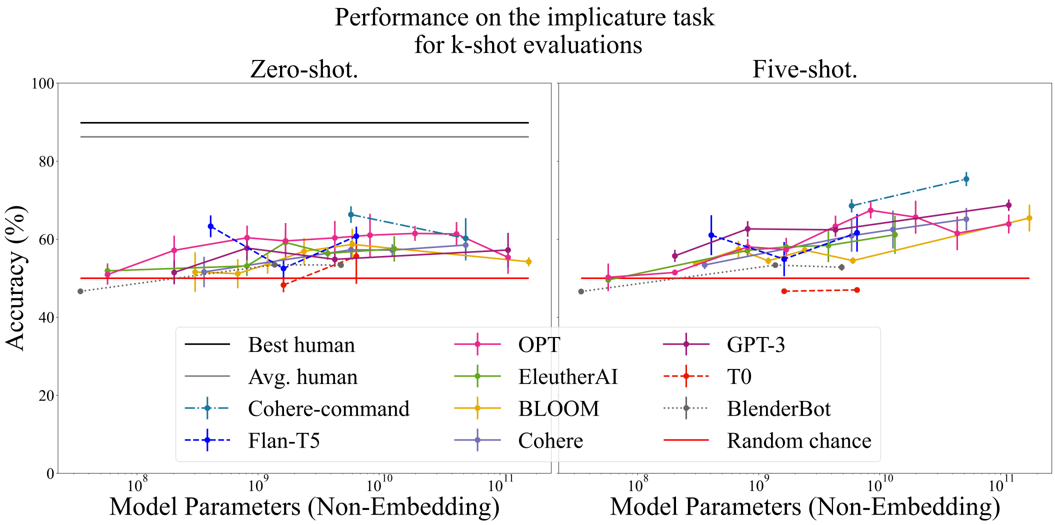


Figure 5.3: Scaling results for the model classes of which we know the number of non-embedding parameters. The error bars show standard deviation over prompt templates. Cohere's command models instruction-tuned at the example-level perform better than all other models. For all models there is still a significant gap between best accuracy and human accuracy.

Table 5.2: Scaling results for OpenAI's text-< engine>-001-series, for which we do not know the number of non-embedding parameters but do know the ordering in terms of size. The colors indicate whether going up in size (from left-to-right) increases performance significantly or not.

Engine	Ada	Babbage	Curie	Davinci
0-shot	56.5% ± 5.8	64.5% ± 1.8 (+8.0%)	69.0% ± 2.9 (+4.5%)	72.3% ± 2.8 (+3.3%)
5-shot	57.6% ± 2.8	66.1% ± 0.3 (+8.5%)	71.3% ± 1.3 (+5.2%)	74.5% ± 1.0 (+4.0%)

to GPT-3-175B and 0.15% per billion parameters for Cohere-command-6B to Cohere-command-52B (exact numbers used to calculate the slope can be found in Appendix D.7.10). If performance is linearly extrapolated from this curve GPT-3 reaches human-level performance at 642b parameters where Cohere-command would need 125b parameters.

Finding 4: GPT-4 reaches average human-level performance with chain-of-thought prompting.

For the model groups that benefit from in-context examples, we attempt to push performance further with chain-of-thought prompting. We manually write a five-shot chain-of-thought prompt for all six prompt templates, and evaluate model performance using this prompt. One of the six chain-of-thought prompts can be found in Table D.4 in Appendix D.2, and the other five are provided

Table 5.3: Results of the chain-of-thought (CoT) experiment for models in the group *Example IT*. The numbers between brackets show the difference in performance with the number on the same row one column to the left. Most models benefit from CoT-prompting, but not all. Additionally, GPT-4 reaches average human-level performance with CoT prompting. ⋆ means size unknown.

Model	0-shot	5-shot	5-shot CoT
text-davinci-001-⋆	72.3% ± 2.8	74.5% ± 1.0 (+2.2%)	67.3% ± 2.6 (-7.2%)
text-davinci-002-⋆	70.6% ± 2.3	79.6% ± 2.0 (+9.0%)	80.1% ± 0.8 (+0.5%)
text-davinci-003-⋆	71.2% ± 2.8	79.7% ± 0.6 (+8.5%)	83.6% ± 0.6 (+4.0%)
ChatGPT-⋆	72.1% ± 6.0	73.9% ± 6.3 (+1.8%)	77.2% ± 1.0 (+3.3%)
GPT-4-⋆	81.8% ± 1.8	82.0% ± 1.7 (+0.2%)	**86.5%** ± **1.0** (+4.5%)
Cohere-command-52B	60.2% ± 5.2	75.4% ± 1.8 (+15.2%)	75.3% ± 0.5 (-0.1%)

in the supplementary material.[7] We only present the results for the group Example IT here, since CoT prompting did not improve performance for two of the base model classes we tried (see Appendix D.7.7). Consequently, we decided not to apply this experiment to the other models in the base group to save compute costs. The results of are shown in Table 5.3. We find that chain-of-thought prompting does not help for all models, but is nonetheless able to boost performance of GPT-4 to 86.5% ± 1.0. This is on-par with average human-level performance, and slightly below human best performance at 89.8%. To illustrate how explicit reasoning helps implicature understanding, we highlight a CoT generated by GPT-4 for an example from the dataset that models persistently get wrong. "A: Is there a bus I can get to the station? B: You can't rely on it". The implicature is yes, there is a bus, you just cannot rely on it. GPT-4 five-shot gets this wrong for all six templates. With CoT it gets it right for five of six templates. The generated CoT for one template is the following:

> Alice says 'You can't rely on it.' Alice must be implying that there is a bus, but it may not be dependable or timely. This means the response to Bob's question is yes, but with a caution about reliability. Answer: yes

More completions can be found in Appendix D.6.

Finding 5: Models often struggle with the same type of examples humans struggle with.

[7] Which can be find at https://openreview.net/forum?id=5bWW9Eop7l.

Table 5.4: An example from the dataset for two types of implicature found in the test set. The rightmost column shows the amount of that type we manually found in the test set.

Type	Example Utterance	Example Response	Impl.	#
Generalised	You know all these people?	Some.	No.	47
Particularised	Want to stay for a nightcap?	I've gotta get up early.	No.	94

We manually labeled 217 examples of the 600 examples in the test set according to a taxonomy. The remaining 383 examples do not fall as clearly within a category and are grouped together as type *other*. In Table 5.4 the two types of examples that occur frequently in the dataset are exemplified. *Generalised* implicatures require little or no context to be understood. They are the simplest type of example in the test set, and generally imply the same thing ("some" almost always implies "not all"). *Particularised* implicatures, by contrast, do require context to be resolved. For example, from Table 5.4, we need the context that it is undesirable to stay up late drinking when one has to get up early (see in Appendix D.1 for more on generalised vs. particularised). In these type of examples, the context needed to resolve it is different every time. We label three other types of implicatures in the dataset, but since the analysis of these examples does not show significant patterns, we present it in Appendix D.7.9. We show the accuracy broken down per example type for two models from the Example IT group, as these patterns hold more broadly for almost all models evaluated (see the detailed results broken down per example type in Appendix D.7.9). Figure 5.4 shows that for lower k, the models often have a significantly worse performance for particularised examples than for generalised examples, just like humans do. For some, like Cohere-command-52B, this is mitigated by few-shot prompting, which brings particularised and generalised performance closer together (sometimes at the cost of generalised performance). For others, like GPT-4, the gap between particularised and generalised performance remains large for all k. From the bottom row in Figure 5.4 we observe that the edge GPT-4 has over Cohere-command-52B seems mostly driven by a higher accuracy on generalised examples. The accuracy on the particularised examples is comparable between those two models.

5.4 Related Work

There is a large body of work that investigates the interplay between pragmatics and computational modeling [Cia+18; SCD20; LRR20; Kim+21; LSD21;

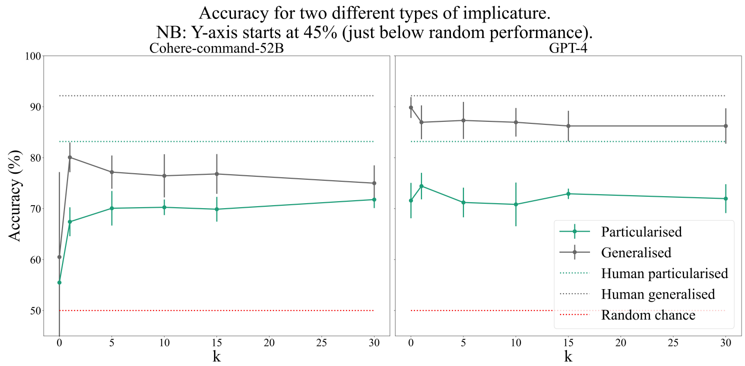


Figure 5.4: The accuracy v. k for the generalised and particularised examples obtained by the Example IT models Cohere-command and GPT-4. Particularised (context-heavy) examples are often significantly more difficult than generalised (context-free) examples for both models and humans. For most models, in-context prompting can mitigate this, but for others (like GPT-4), a significant gap remains. We see that Cohere-command-52B achieves similar performance as GPT-4 on the particularised examples, but significantly lower on the generalised examples.

Jer+20; Par+21; Hos+23]. [Cia+18] introduce the task of predicting adverbial presupposition triggers, which are words like 'again' that trigger the unspoken presupposition that an event has happened before. [SCD20] study the ability of computational models to do scalar inferences, finding that models use linguistic features to make pragmatic inferences. [Kim+21] find that a substantial part of question-answering datasets contain questions that are unanswerable due to false presuppositions (i.e. "which linguist invented the lightbulb"). [Hos+23] present a dataset for selecting entities with indirect answers, and find that language models adapted for this task get reasonable accuracy, but that there is room for improvement. The difference with this body of work and ours is that we look at the emergence of pragmatic understanding from large-scale language modeling. [Jer+20; Par+21] are early works investigating the emergence of pragmatic understanding in pretrained language models, but they only look at scalar implicatures and presuppositions. [Zhe+21] are the first to evaluate pretrained language models on conversational implicatures. This is important pioneering work highlighting the difficulty of implicature for language models, but their evaluations require task-specific training and the models they evaluate are relatively small. In contrast, our evaluation protocol is applicable out-of-the-box and is much more comprehensive, evaluating models up to 176 billion parameters and using in-context prompting.

Additionally, [Zhe+21] benchmark synthetic data whereas this work evaluates performance on naturally occurring implicatures [GM20]. We believe this to be a better representation of the true distribution of implicatures in natural dialogue.

The standard set of benchmarks LLMs are evaluated on covers many tasks, but even though implicature is one of the most important aspects of language pragmatics [Lev83], it is only evaluated as part of BIG-bench [Sri+22]. Unfortunately, the methodology used by the BIG-bench implicature task contributors has limitations, which call into question the validity of their claims. Firstly, the task contributors discard a subset of the data that is ambiguous according to them. In our view this defeats the point of the benchmark. Implicatures are a type of non-literal, ambiguous language the intended meaning of which humans often easily interpret; comparing the way humans and models do this is precisely what we are interested in. In turn, we expect performance on the BIG-bench task to overestimate the ability of LLMs to resolve naturally occurring implicatures. We keep this challenging subset of the data and instead use human evaluation to deal with examples that are too ambiguous to understand. Secondly, the difference in performance between their average and best rater is 18%, whereas for our evaluations this difference is 6%. This indicates their human evaluation is of low quality, but it is impossible to verify because there are no details available on how the annotation is done. Finally, BIG-bench uses only base LLMs and no SotA fine-tuning methods. In summary, we use a more challenging dataset, and in turn at least six times more evaluations per model, we provide higher-quality human annotations, and evaluate four different categories of LLMs to investigate which aspects of LLMs contribute to their performance on implicature understanding.

5.5 Discussion, Limitations, and Future Work

In this study we use prompting to evaluate whether different groups of LLMs can resolve implicatures. In designing our experimental protocol, we carefully considered various alternatives, and here we discuss limitations of the chosen approach.

Firstly, evaluating LLM competencies is inherently uncertain and sensitive to prompt choice. Nonetheless, we are confident our evaluation is comprehensive enough to assess implicature understanding: we apply six different prompt templates per test example, each used in three different prompting techniques (zero-shot, few-shot, chain-of-thought). Additionally, in the appendix we present alternative zero-shot prompts and task specifications (Appendix D.7.3 and D.7.1 respectively), but since these did not improve performance they were not further considered.

Another limitation is the fact that a subset of the models we evaluate are behind APIs. This means models are subject to change (affecting reproducibility) and certain details about these models are unknown. This affects the group instruction-tuned at the example-level, which is the group we find outperforms all others and has the most favourable scaling properties. How do we know instruction-tuning at the example-level is the main driver behind these findings without controlled A/B testing? Unfortunately, due to the secrecy surrounding the exact implementation of these models we cannot be certain, but we can be relatively confident. We evaluated ten models across six model classes and two APIs in the group example-level instruction tuned. Within this group, models probably vary significantly in other training and architecture details (especially Cohere-command models versus OpenAI models). The most salient commonality they share with each other and none of the other models is multi-task instruction-tuning at the example level, making it likely that this is the driving factor of their performance. A further datapoint in favour of this conclusion can be seen in Figure 5.3 (right); base models at similar scales as Example IT models perform significantly worse. We see that Cohere-command 52B significantly outperforms Cohere-base 52B, and the only difference between those models is instruction-tuning at the example level (Cohere-command is fine-tuned from Cohere-base). In fact, Cohere-command 52B outperforms other base models more than 3 times the size by a large margin (e.g. GPT-3 175B, BLOOM-176B, OPT-175B). We are therefore confident that instruction-tuning at the example-level is important for pragmatic understanding, a finding which can guide the development of open-source models capable of pragmatic understanding. Investigating the exact effect of this type of instruction-tuning on pragmatic understanding in a controlled setting is an interesting future work direction (e.g. by isolating the effect of data diversity from instructions).

A further limitation is that some evaluations are subject to API stochasticity, which we address in Appendix D.7.5. After running the zero-shot experiment ten times through each API we conclude there is some stochasticity, but it is too small to impact our conclusions. We publish exact timestamps at which we queried APIs in Appendix D.8. Further, a downside of doing a comprehensive analysis on many models is compute costs. In Appendix D.9 we publish a list of exact compute used (time and hardware), as well as estimated carbon emissions for each of the models that are not behind an API.

Finally, the likelihood ranking approach we take limits our study to implicatures with clear alternative. However, implicatures in natural language can entail more complex propositions. For example, imagine Esther now asking "Can I use your stapler?" and Juan responding "Here's the key to my office.". Juan is implicating that (1) Esther can use the stapler, (2) the stapler is located in the office, and (3) the office is currently locked. This leaves ample room for the design of benchmarks with implicatures entailing multiple non-binary propositions.

Using our protocol that evaluates LLMs on binary implicature resolution, we establish a significant gap with human understanding for SotA LLMs in three categories; large-scale pre-trained models, models fine-tuned on conversations, and models fine-tuned with benchmark-level instructions. By contrast, we find that models fine-tuned on example-level instructions perform significantly better. This group also exhibits the best correlation between accuracy and model size. Scaling analysis shows that for some large-scale pre-trained models accuracy also positively correlates with model size, but the best model in this group would need at least five times more parameters to reach similar performance. From these results, we conclude that instruction-tuning at the example level is important for pragmatic understanding. We hypothesise that there is something about the multi-task data diversity obtained from example-level instructions (i.e. each example a new task) that makes pragmatic understanding appear at smaller scale.

Pragmatic understanding is intimately linked to theory of mind in human development, with researchers arguing that children's ability to understand the mental states of others provides the cognitive foundation for interpreting conversational implicatures [MAD07]. This raises a key question: given the results in this chapter, have models similarly developed the theory-of-mind capabilities that precede pragmatic understanding in children? The next chapter examines the earliest theory-of-mind-like behaviours observed in infants to explore this question.

Chapter 6

A Case Study in Social Reasoning: Theory of Mind

6.1 Overview

Given that the previous chapter establishes an understanding of conversational implicatures at a human-level, one may wonder whether the precursors to this type of pragmatic understanding in human development have similarly emerged in models. In this chapter, we continue the characterisation of the social reasoning abilities of LLMs by investigating a key socio-cognitive foundation for pragmatic reasoning: theory of mind (ToM). Theory of mind is the ability to reason about unobserved mental states of other agents. It is considered central to many aspects of human cognition, like linguistic communication [MAD07]. Recent studies on the emergence of ToM in LLMs yield conflicting results; some works suggest it has emerged [Kos24; MH23], and others suggest it has not [Ull23b] or at least not at a level comparable to humans [Tro+22; Sap+22; Sha+23]. While Kosinski [Kos24] shows certain LLMs can pass classic false-belief tests, Ullman [Ull23b] demonstrates that those same models fail on minimal alterations to the tasks that change the expected answer.[1] This evidence suggests models memorise training patterns without actually mentalising. However, the evaluated tasks are classic false-belief tasks that are abundant in the pre-training data, and it remains an open question whether models can mentalise in situations that are less likely to occur in their training data.

[1] Ullman tests LLMs on unexpected contents tasks where the contents are in see-through containers, altering the answer to the false-belief tests.

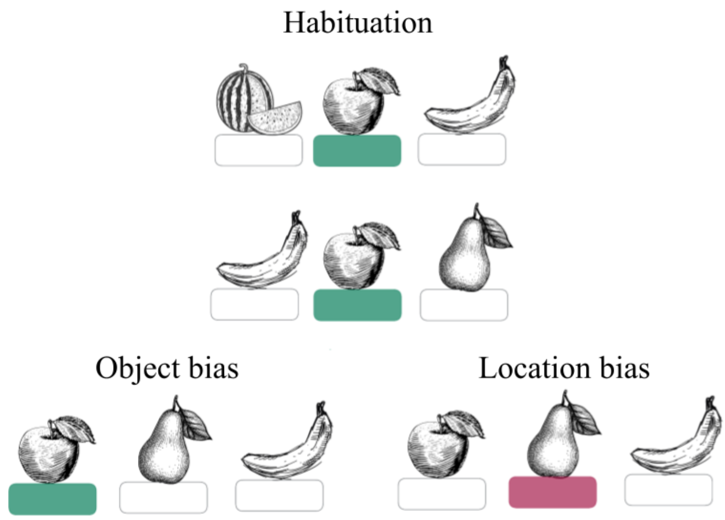


Figure 6.1: A visual depiction of our test inspired by Woodward [Woo98]. We prompt an LLM with k ambiguous linguistic *habituations* that can be explained either by the goal being the object or the location ($k = 2$ in the image). We then test the bias the model shows for assuming the goal was the object (left-bottom) or the location (right-bottom). We say a model *selectively* encodes the goal if it shows a distinct bias when an agent appears to be acting purposefully.

How can we investigate machine theory of mind in models that have seen all the classic tasks from developmental psychology and regurgitate their patterns even when these tasks are worded differently? In this chapter, we avoid pre-existing text-based tasks and look at one of the developmentally earliest occurrences of theory-of-mind-like human biases: selectively encoding the goal object of an actor's reach [Woo98]. In her seminal study, Woodward shows that pre-linguistic infants exhibit a bias for encoding an agent's goal object over a goal location (detailed in Section 2.1.1). Similarly, we ask the question: *do large language models selectively encode the goal object of an actor's reach?* We say a model passes the test if it shows a distinct bias between the agent acting purposefully and otherwise (see Figure 6.1). For a behaviour to be considered theory of mind, the same behaviour should not show up when the task does not involve a goal-directed agent [FF12; DHD14].

Our results show that GPT-3.5-turbo and GPT-4 pass the criterion for saying that they selectively encode the goal of an actor's reach for some of our prompt variations, but not for other semantically equivalent ones. These results contribute to the picture from existing work on ToM in LLMs, concluding that even the developmentally earliest ToM-like human behaviour does not robustly

Table 6.1: The prompt variations we use in our evaluations. For each template text, the target word is **bolded**.

Template variation	Test case	Example of differing template part
Fruit targets	Animate Inanimate Control	Wendy grasps the **kiwi** A rod moves to and touches the **kiwi** Wendy accidentally touches the **kiwi**
Fruit targets (anim)	Animate Inanimate Control	A person named Wendy grasps the **kiwi** An inanimate rod moves to and touches the **kiwi** A person named Wendy accidentally touches the **kiwi**
Pillar targets	Animate Inanimate Control	Wendy grasps the item on the **first** pillar A rod moves to and touches the item on the **first** pillar Wendy accidentally grasps the item on the **first** pillar
Pillar targets (anim)	Animate Inanimate Control	A person named Wendy grasps the item on the **first** pillar An inanimate rod moves to and touches the item on the **first** pillar A person named Wendy accidentally grasps the item on the **first** pillar

show up in current SotA LLMs. Our findings further highlight the importance of designing multiple prompt variations for each task: depending on how the task is framed, conclusions can be opposite.

6.2 Evaluation Protocol

In this section we outline the method we use to answer the question: *do language models selectively encode the goal of an actor's reach?*

We prompt a set of LLMs with *habituations* that can be explained both by the goal of an actor's reach being an object, as well as a location. We then look at whether LLMs exhibit a bias for assuming the goal is the object or the location (see Figure 6.1). We investigate the bias the model shows in three situations: an agent is purposefully reaching for an object, an inanimate object moves and touches an object, and an agent is acting accidentally and touches an object.

Defining object and location bias. We want to investigate the question whether models store knowledge that leads them to encode the goal-related properties of an agent's reaching event, and that this knowledge does not get encoded in similar events involving inanimate objects. To this end, we design the following test cases: an animate test case where the prompt contains k habituations in which an agent reaches for the same object in the same location. A test case is appended to this prompt where the goal object is placed in a different location. We then obtain the likelihoods the model assigns to continuing the full prompt as if the same location with a novel object is reached for by the agent (*location bias*), or the same object at a different location (*object bias*, see Figure 6.1). Below is an example for an agent, Wendy, who has a preference for kiwis, with $k = 2$ habituations:

> There is a kiwi on the first pillar, an orange on the second pillar, and a fig on the third pillar. Wendy grasps the item on the first pillar.
> There is a kiwi on the first pillar, a fig on the second pillar, and an orange on the third pillar. Wendy grasps the item on the first pillar.
> There is an orange on the first pillar, a kiwi on the second pillar, and a fig on the third pillar. Wendy grasps the item on the *first/second*

In this example, a model that assigns a higher probability to *first* is said to exhibit a location bias, whereas a model that assigns a higher probability to *second* exhibits object bias. Independently, we test the model on the same example with an inanimate object:

> There is a kiwi on the first pillar, an orange on the second pillar, and a fig on the third pillar. A pole moves to and touches the item on the first pillar.
> There is a kiwi on the first pillar, a fig on the second pillar, and an orange on the third pillar. A pole moves to and touches the item on the first pillar.
> There is an orange on the first pillar, a kiwi on the second pillar, and a fig on the third pillar. A pole moves to and touches the item on the *first/second*

We generate 100 examples with a roughly equal distribution over object and location targets (in this example template, the targets can be one of "first", "second", and "third"). We define the *object bias* o_b as the conditional probability that the object bias target is chosen by a model given that the model has to either choose the object or location bias target, as in

$$o_b = \frac{p(\text{object bias target})}{p(\text{object bias target}) + p(\text{location bias target})}, \tag{6.1}$$

where each probability $p(\cdot)$ is conditioned on the prompt like $p(\cdot \mid \text{prompt})$.

In some cases we do not have access to the probabilities assigned to each target by a model (i.e. GPT-3.5-turbo and GPT-4 have restrictive APIs). Instead, we sample those models ten times for each prompt with a temperature of 1, recording how often they output the object bias target c_o (*second* in the previous example) or the location bias target c_l (*first* in the previous example). Using these counts, we estimate the object bias o_b of a model for each example as the fraction of times it chooses the object bias target:

$$\hat{o}_b = \frac{c_o}{c_o + c_l} \tag{6.2}$$

We discard all samples where a model does not choose the object or location bias target and record them separately as unclassified in the c_u count. We report summary statistics for the obtained probabilities and the counts c_o, c_l, and c_u for each model and prompt template in Appendix E.1.

The criterion for selective encoding. We add a control task where the agent accidentally reaches for the item, meaning that the object is no longer the agent's goal. We do this by slightly changing the animate prompts. For example in one template we change *Wendy grasps the item* ... to *Wendy falls and accidentally grasps the item* .... Note that although this is similar in spirit to [Ull23b], the difference is that we show the model multiple (k) habituations with the same change.

The criterion for saying that a model selectively encodes the goal of an actor's reach is if it exhibits a distinct bias in the animate case compared with the bias shown in the inanimate and control case. In other words, the bias in the animate case should be different from the bias in the inanimate and control case, and the latter two should be similar. If this criterion is passed, it means the model has a different bias when there is a goal-directed agent involved than when there is an inanimate or non-goal-directed agent involved. Besides the selective encoding of the goal, we can also contrast the specific bias the model demonstrates with human infants, who show an object bias in the animate case, and no bias in the inanimate case in Woodward's visual test (infants are not tested with a control task).

Prompt variations. We vary the agent names, pillar fruits, and inanimate objects to get a larger set of test examples (namely 100 per test case). Additionally, for each test case we design a set of four different prompts, to test for things like irrelevant alterations of the text. The first prompt has already been

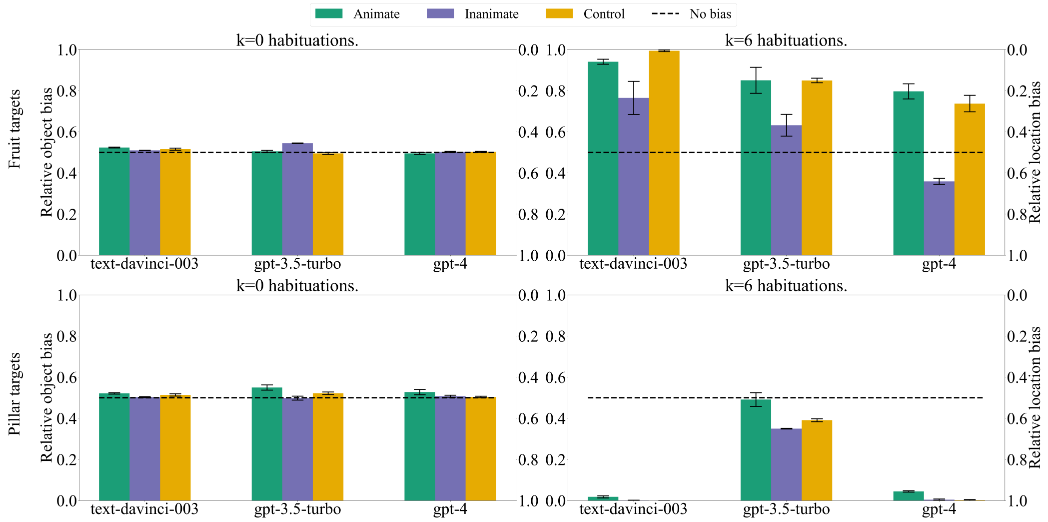


Figure 6.2: The results for text-davinci-003, GPT-3.5-turbo, and GPT-4 for $k = 0$ (left) and $k = 6$ (right) habituations. For $k = 0$, we expect the object bias to be roughly 0.5 (equal selection of object bias target and location bias target). For $k = 6$ in the right column of the figure, recall that if the model is encoding irrelevant surface-level patterns of the prompt, we expect a strong object bias for the fruit target prompt variations (top-right) and a strong location bias for the pillar target prompt variations (bottom-right), regardless of whether the test case is animate, inanimate, or control. Indeed, we observe a general stronger object bias for the top row than the bottom row when $k = 6$. We further see that all models have a higher object bias for the animate test cases than for the inanimate, but show a similar bias for the control test case as the animate case for the fruit target variations (top-right plot). GPT-3.5-turbo and GPT-4 are the only models that also show a similar bias for the control test case as the inanimate case, which means they selectively encode the goal of an agent's reach (i.e. the biases for inanimate and control are similar and distinct from animate). However, only when the target is the pillar (bottom-right plot), and GPT-4 does so only very weakly. The error bars represent the standard deviation over the two prompt templates in each group (fruit targets and pillar targets).

presented in this section. This prompt is of the type *pillar target*, because the target on which the model is evaluated is a pillar choice (first, second, or third). In the second prompt the target is not the pillar location, but the fruit itself (e.g. replace *Wendy grasps the item on the first pillar* with *Wendy grasps the kiwi*), and so the prompt is of the type *fruit target*. For both of these prompts, we also construct a variation in which we explicitly denote that the agent is animate and the inanimate object is not (e.g. replace *A pole moves to ...* with *An inanimate pole moves to ...* and replace *Wendy grasps ...* with *A person* *named Wendy grasps ...*). This leaves us with four prompt variations in total, which are fully presented in Table 6.1. Templates only differ in the sentences describing the agent's reach, otherwise they share the pattern previously shown.

Note that for the pillar target prompt variations, a prompt with multiple habituations repeats the action of reaching for the same pillar multiple times (e.g. *grasps the item on the first pillar*). Hence, a language model that is sensitive to surface-level patterns in text might put a high probability on the same pillar from the habituations to complete the test case phrase *grasps the item on the* _, which would result in a recorded location bias for this model. This is why we construct the prompt variations where the target is the fruit instead (*Fruit targets* in Table 6.1). In those variations, a prompt with multiple habituations repeats the action of grasping the fruit (e.g. *grasps the kiwi*). This might cause a language model to put high probability on the same object from the habituations to complete the test case phrase *grasps the* _, in which case an object bias would be encoded. Therefore, if the model is not encoding the semantics of the prompt and simply repeats surface-level patterns, we expect an object bias for the fruit target variations, and a location bias for the pillar target variations, regardless of whether the test case is animate, inanimate, or control.

This protocol has several benefits over other approaches of investigating machine ToM from literature. Firstly, the underlying task logic is visually presented to pre-linguistic human infants in literature, making it less likely that the exact task appears in the training data of pre-trained language models. Nonetheless, the reasoning pattern might be numerously described. In similar spirit to Ullman [Ull23b], we extend Woodward [Woo98] by adding a control task where the agent acts accidentally, nullifying the assumption that the agent is acting in a goal-directed way. Like the inanimate case, the object bias should not show up in this control task. Another benefit is the habituations that are reminiscent of few-shot prompting in LLMs [Bro+20a], but unlike true few-shot examples these do not leak any information about the expected output. These examples both serve to habituate a model in order to probe for a bias, as well as to guide the model to the task. Importantly however, even though we can use our protocol to make empirically backed claims about whether or not LLMs selectively encode the goals of agents, we can make no statements about how the model does it and whether there is reasoning involved. Similarly, Woodward makes no assumptions about what kind of knowledge infants use to encode the goal object of an actor's reach; she just shows that they do.

6.3 Experiments

We evaluate three different models on our test cases, all of which are OpenAI API models (text-davinci-003, GPT-3.5-turbo, and GPT-4). For the latter two, we do not have access to their likelihoods; to obtain an estimate despite this we apply a sampling strategy as described in Section 6.2. The results are presented in Figure 6.2, and the numbers underlying this figure are presented in Appendix E.1. The left column in Figure 6.2 shows the results for $k = 0$ habituations, which is a sanity check that the model does not have a strong bias for a target a priori. These numbers should ideally show no bias (0.5 object bias and location bias), which is roughly the case. Below, we discuss the results for $k = 6$ habituations, which is the number of habituations [Woo98] uses with infants.

Finding 1: All models show a stronger object bias in the animate case than in the inanimate case, but only GPT-3.5-turbo and GPT-4 selectively encode the goal of an agent's reach, and only for two of the four prompt variations.

In general for all three test cases (i.e. animate, inanimate, control), we see a stronger object bias for the fruit target variations (top-right in Figure 6.2), and a stronger location bias for the pillar target variations (bottom-right in Figure 6.2). As mentioned at the end of Section 6.2, this is unsurprising given the repeated patterns in the habituations, and any deviation from this pattern is notable and points to encoding of semantics over surface-level properties. All models show a higher object bias in the animate case than the inanimate case, which is similar to the effect that Woodward finds for infants (object bias in the animate case, no bias in the inanimate case). The only models which pass the criterion for saying they selectively encode the goal of an agent's reach on our test set are GPT-3.5-turbo and GPT-4 (recall that the criterion is a strong difference in bias for the animate test case compared to the inanimate and control cases). However, the criterion is primarily passed for the two prompts where the targets are the pillars instead of the fruits (bottom-right), and only very weakly in GPT-4's case.

Finding 2: text-davinci-003 does not appear to selectively encode the goal of an agent's reach.
Although text-davinci-003 shows a stronger object bias in the animate case than the inanimate case, it shows the same bias as the animate case in the control test case (overlapping error bars for the pillar targets in the bottom-right plot). This means we cannot say the model selectively encodes the goal of an agent's reach, because it encodes text similarly when an agent is acting purposefully as when the agent is not acting in a goal-directed fashion. Looking at the magnitude of the biases again, we see that text-davinci-003 shows a strong object bias for the fruit target templates, whereas it shows a full location bias for the pillar target templates. For the latter, it might simply be using the heuristic of repeating the pillar from habituations.

Finding 3: All three models are heavily influenced by semantically irrelevant alterations of the prompt, but are clearly not only encoding surface-level statistics of the text.
Comparing the top-right and bottom-right plots in Figure 6.2, we find that all three models show much more location bias when the target is the pillar instead of the fruit. However, it is not the case that the models simply have an object bias when the target is the fruit and a location bias when the target is the pillar. Although this shows that the models' internal reasoning can be heavily influenced by superficial differences in output requirements, the strong biases that go against the surface-level repetitions do indicate encoding of the semantics of the text.

6.4 Related Work

Recently, classic ToM tests from developmental psychology have been extensively applied to LLMs. However, these studies have conflicting results. Kosinski [Kos24] claims theory of mind has emerged in a subset of OpenAI's API models, but the evaluation protocol has been pointed out as flawed by Ullman [Ull23b]. Similarly, Sap et al. [Sap+22] show that GPT-3 achieves well below human performance on a range of different ToM tasks. The methodology used in that study is however critiqued by Moghaddam and Honey [MH23], who apply similar tests but use SotA prompting techniques and show that OpenAI's models that are fine-tuned with RLHF achieve human-level performance on the ToM tasks. By contrast, Shapira et al. [Sha+23] show that LLMs can robustly solve some ToM tasks, but not others, and conclude that models have some ToM capabilities, but that these are not robust. Concurrently and motivated similarly to our work, Gandhi et al. [Gan+23] propose a procedurally generated benchmark testing for false-belief tasks in natural-sounding situations. They find GPT-4 shows human-like ToM inference patterns, but less robustly than humans do.

Woodward [Woo98] conducts her study with the aim of exploring how infants perceive and comprehend others' actions. The study focuses on investigating infants' ability to selectively encode the goal object of an actor's reach. Drawing inspiration from Woodward [Woo98], Gandhi et al. [Gan+21] apply a similar task to neural networks, aiming to determine whether machines can represent an agent's preferred goal object. However, to our knowledge, there is currently no study that applies the task from Woodward specifically to pre-trained LLMs.

6.5 Discussion, Limitations, and Future Work

Our results show that the tested LLMs do not robustly encode the goal-related properties of an agent's reaching action. GPT-3.5-turbo and GPT-4 do treat text differently when there is a goal-directed agent involved, but do not do this equally for semantically equivalent prompt variations. Additionally, the biases they show are very different from the bias human infants show in Woodward [Woo98]. The specific bias we investigate is very basic, appearing in infants as young as six months old. Our results indicate that ToM-like human biases might not emerge from large-scale pre-training on text or instruction fine-tuning, at least not in the way we might expect them to. This suggests that studies investigating the emergence of ToM in LLMs should not expect a machine ToM that is comparable to human ToM, but should instead focus on identifying in what way machines reason about the mental states of others, forming a machine theory of mind. Additionally, our results show that studies need to take into account the sensitivity of models to semantically irrelevant surface-level patterns in text, which might be very different from humans' responses to such patterns. In our study we deal with this by designing prompt variations that would result in an opposite effect if only surface-level patterns are encoded. Any deviation from this pattern indicates encoding of semantics over irrelevant patterns. Our results serve as a first step towards comparing human theory of mind and machine theory of mind without preconceived notions of the kind of mentalising the machine should do.

We take the approach of linguistically presenting a ToM test to LLMs that is traditionally only tested *visually* in pre-linguistic infants. Although we view this as a strength of the protocol because it makes it less likely that the test appears in the training data, it also means that a lack of human-like bias in LLMs may simply indicate that this bias does not show up linguistically. To say LLMs show a different bias than humans in this task, we need to administer the same tests to human adults. In future work, we want to conduct human evaluations on our linguistic test to identify the biases humans show.

One hypothesis for why selectively encoding the goal object of an actor's reach has not yet emerged is that learning such a bias might simply not be consistently useful for next-token prediction in pre-training on text. Another hypothesis is that pre-training on large-scale internet data representing too many agents with noisy beliefs hindered the ToM-like ability [And22]. An interesting direction for future work would be to test if fine-tuning pre-trained models on data reflecting agent preferences for objects, and random reaching events for inanimate objects can lead to the emergence of ToM-like ability. Successful next-token prediction on this dataset requires inferring the underlying agent preferences of the agents that occur in the data, as well as learning that inanimate objects have no preferences. Using this protocol, we can control how consistently useful the object bias is for next-token prediction by adding noise to the data, and seeing how this affects the resulting biases in the model for novel agents and objects.

Our evaluation protocol opens up further interesting avenues for future work. Although prior work in machine ToM mostly views it as a static ability that you can either have or not, current approaches to ToM in humans and other animals recognize that mentalising inferences are dynamic [Bak+17] and graded in performance [DHD14]. These insights have recently been applied to make progress on the Baby Intuitions Benchmark [Gan+21] by applying a Bayesian hierarchical framework [Lan00]. Since our evaluation protocol allows varying the number of habituations, future work might take a similar approach, and investigate how varying degree of observations change the model's predictions of an agent's behavior, as the studies investigating human ToM did [DHD14; BST09; Bak+17; SGG14; YDF08]. For example, repeated trials of hide and seek [DHD14] can differentiate ToM abilities in different clinical populations [dDD20] and even across primate species [Dev+17]. Models taking this approach successfully generate precise quantitative predictions of how people infer preferences and beliefs of other agents over a range of parametrically controlled stimuli [Bak+17].

In this chapter, we introduce a new evaluation protocol to test large language models' (LLMs) capabilities in the context of Theory of Mind (ToM). Inspired by Woodward [Woo98], we prompt LLMs with ambiguous examples of agents interacting with objects. We let the models predict the agent's next interaction, which can be either explained as an explicit agent goal in terms of location or object choice, or by random chance—allowing us to assess if *a model selectively encodes the goal of an agent's reach*. Extending the original study, we do not only test against inanimate interactions but also use a control task with accidental interactions. This addition appears crucial, as without it our results would have concluded LLMs selectively encode the goal object of an agent's reach, whereas the recorded biases in the control test case call into question this conclusion. We further show that all models are highly susceptible to be influenced by minor prompt variations that do not semantically change the task. These findings serve as a cautionary tale for researchers investigating human-like biases in LLMs; careful evaluations should be designed in order to control for models repeating patterns from training data without robustly demonstrating the bias of interest. Moreover, our findings indicate that the social reasoning abilities of LLMs may not follow a similar developmental trajectory as humans' abilities.

Chapter 7

Conclusions

This chapter concludes the investigation into how LLMs perform reasoning tasks. I begin by summarising the four main ways in which this thesis characterises LLM reasoning. I then discuss the broader implications of these findings, situating them within the wider field, and speculate on where the compute scaling paradigm may lead. Finally, I outline concrete future directions arising from each of the preceding four chapters.

7.1 The Nature of LLM Reasoning

1. LLM reasoning is markedly different than retrieving facts.
Through a large-scale study ranking five million pre-training documents according to their influence on model completions, I quantitatively and qualitatively demonstrate that models develop distinctly different strategies for reasoning versus factual retrieval. Even when documents with answers to the exact reasoning steps are present in the pre-training data, these are ranked typically much less high than the documents with answers to the factual retrieval completions. Instead, models rely on what I term *procedural knowledge* in pre-training; knowledge that is useful for multiple instances of a reasoning task. The influence analysis further reveals two key distinctions between reasoning and factual retrieval. First, models depend less heavily on individual pre-training documents for reasoning than for factual tasks, as measured by influence scores representing how much including each document would decrease loss on completions. Second, influence scores for reasoning show less volatility than those for factual retrieval, indicating that reasoning draws from a more stable and broadly available set of training data. These convergent findings demonstrate that models acquire generalisable procedural knowledge during pre-training, employing a strategy for reasoning that extends beyond

memorisation.

2. LLMs have acquired reusable computational principles that abstract away from inputs.
In a follow-up study, we auto-regressively fine-tune LLMs on previously unseen program source codes, and demonstrate that the ability to evaluate these programs for inputs emerges. This indicates that over the course of their pre- and post-training stages, LLMs have acquired computational principles that enable them to learn diverse emergent capabilities from further auto-regressive training on symbolic source code. For example, when we first fine-tune models to predict outputs for inputs to common programming challenges in Leetcode and then fine-tune them to auto-regressively predict the tokens of source code for novel custom cipher algorithms, the ability to evaluate these ciphers for inputs also emerges. This transfer indicates that while LLMs are trained on the surface task of predicting the next token, this process somehow leads them to internalise computational principles which they can apply to novel situations at inference time. Fine-tuning models on programs in this way leads to a more uniform performance across inputs than when training on input-output pairs that mirror naturally occurring data. In fact, we demonstrate that fine-tuning models on a single piece of cipher code augmented with different prompts without changing the source code itself leads to uniform and non-trivial performance across inputs. Perhaps most strikingly, the most capable model we fine-tune to auto-regressively ingest two separate program's source codes with next token-prediction subsequently shows some ability to evaluate compositions of these for inputs entirely within its weights (i.e. without outputting intermediate computations in a chain of thought). These findings confirm that models indeed acquire abstract procedural knowledge from training which can be applied to unseen context at test time.

3. LLMs learn to infer communicative intent implicit in ambiguous language from large-scale auto-regressive pre-training.
We investigate this by comparing how humans and different types of LLMs resolve conversational implicatures; a type of language where conversational turns imply more than their literal meaning. For example, when asked "Can you come to my party tonight?" responding "I'm sick" implies "no" without explicitly stating it. Our findings show that the ability to correctly resolve such binary conversational implicatures emerges in base models and improves with

scale. However, the most substantial gains occur when models are post-trained on human-written instruction-completion pairs. Using chain-of-through prompting, the most capable model we evaluate achieves human-level performance at implicature understanding. The scaling trends suggest that human-level implicature resolution could emerge from parameter scaling of base models alone, but high-quality post-training methods produce more efficient pragmatic reasoners at current scales.

4. LLMs do not robustly encode text differently based on whether an intentional agent is involved.
Given that LLMs demonstrate human-level pragmatic reasoning abilities, we investigate whether they have also developed the foundational socio-cognitive skills that enable such reasoning in humans. Specifically, we test whether state-of-the-art LLMs exhibit theory-of-mind-like behaviours analogous to those observed in early human development. Infants as young as 6–9 months expect agents to have preferences that inanimate actors do not have, demonstrating early theory of mind development. We adapt a visual experiment conducted on pre-linguistic infants into a textual format suitable for LLMs. Crucially, we include an experimental control condition featuring animate agents acting accidentally, which eliminates any intentional preference hypothesis. Our analysis examines whether LLMs process text systematically differently for intentional animate agents compared to both inanimate objects and accidentally-acting agents, finding no consistent differences across these conditions. Notably, without the accidental control condition, we would have incorrectly concluded that LLMs do encode agent animacy and intentionality, highlighting the critical importance of rigorous experimental controls when adapting psychological experiments for AI evaluation.

Taken together, my findings characterise LLM reasoning not as the retrieval of memorised facts, but as the synthesis of procedural knowledge during pre-training. Models appear able to extract reasoning principles implicit in their training data and apply them to diverse questions. Auto-regressive next-token prediction further supports the acquisition of knowledge at multiple levels of abstraction, which can then be applied to novel contexts at inference time. Beyond formal reasoning, some social reasoning abilities that human children acquire in direct social contexts also emerge from large-scale next-token prediction. Yet LLMs' social reasoning does not necessarily follow the same developmental trajectory as humans, suggesting it should be characterised on its own terms, without relying on expectations from developmental psychology.

7.2 Discussion

I set out to understand how models trained on trillions of tokens of sequential data learn to perform reasoning tasks. Amid the widespread adoption of LLMs, a fundamental question remained unanswered: can these systems generalise beyond their training distributions in ways that might contribute genuinely novel knowledge, or are they fundamentally constrained by the patterns they encountered during pre-training? I cannot claim to have definitively answered this question. However, my understanding of LLM reasoning has significantly deepened throughout my doctoral studies. In this chapter, I discuss the implications of my research, draw some tentative conclusions, and outline its limitations.

7.2.1 Stochastic Parrots, or Not?

The idea that training LLMs with the simple task of predicting the next token leads them to internalise more fundamental computational principles, like logical reasoning patterns and causal relationships, seemed outrageous to many besides perhaps a small group of researchers at OpenAI. While models were rapidly saturating reasoning benchmarks, sceptics (including myself) claimed they are simply regurgitating what they have seen before. The findings in this thesis show that this is at least not true at the level of verbatim regurgitation of answers to reasoning steps from pre-training data. However, critics then often point out that LLM reasoning does not *verbatim* regurgitate what it has seen before, but extracts information from vast distributed training data in uninterpretable ways and recombines this to seem like proper reasoning, but that would not generalise to truly unseen situations. This is a more difficult position to argue against, as it is vague enough to cover most forms of generalisation. However, the level of emergent reasoning demonstrated in this thesis seems far from limited to shallow training patterns.

In Chapter 4 we demonstrated in a controlled setting what Chapter 3 already hinted at in a more messy large-scale investigation of pre-training: models have acquired seemingly symbolic computational skills from training that they can reuse for unseen applications. The most complex generalisation we observe is while auto-regressively ingesting pieces of code, the ability to perform in-weight evaluation of compositions of these for inputs emerges. This requires parametric retrieval of two programs from disparate pieces of training data and evaluating each in sequence for inputs, entirely within weights (i.e. out-of-context). All while never having been trained on input-output pairs for either of these programs, let alone their composition. Of course, each operation that make up these programs are highly familiar to the models (simple arithmetic and control flow), and if we were to give the source codes in context the model could easily step through it to get the answer. The point of these experiments is not so much demonstrating LLMs' arithmetic skills, but rather demonstrate the emergent acquisitions of knowledge at different levels of abstractions from simple generative training on source code.

7.2.2 The Surprising Effectiveness of Next-token Prediction at Scale

Beyond the controlled fine-tuning experiments described in Chapter 4, several key findings in this thesis demonstrate that next-token prediction at scale serves as a remarkably powerful learning objective. Chapter 3 reveals how fundamentally different learning patterns emerge during pre-training using the same self-supervised objective across large-scale text and code datasets. This supports the foundational claim that language models function as unsupervised multitask learners [Rad+19]. More strikingly, this passive text-based training enables models to infer implicit communicative intent, providing direct empirical evidence against theories arguing such capabilities cannot emerge from passive text learning alone [BK20]. Chapter 5 shows that while human-level pragmatic performance requires post-training through opaque methods (potentially including reinforcement learning), base models nevertheless exhibit meaningful pragmatic understanding that scales with model size. These empirical results align with the theoretical framework proposed by Merrill, Warstadt, and Linzen [MWL22], which suggests that perfect mastery of the language modelling objective necessarily means the acquisition of complete entailment semantics. Our findings provide concrete evidence for this theoretical position.

7.2.3 Pre-training or Post-training, Where do Models Learn to Reason?

The findings above support the conventional view that model capabilities emerge during pre-training, with post-training serving primarily to make these capabilities more accessible and aligned with human values. This perspective finds support in several studies demonstrating that supervised fine-tuning enhances rather than creates underlying abilities [Jai+24; KSR24; Pra+24]. However, my own research presents a different picture. In the published work underlying Chapter 5, I argue that supervised fine-tuning on human-written instruction-completion pairs is important for pragmatic understanding, given the dramatic performance improvements we observe for instruction-tuned base models.

Two years later, my thinking has evolved toward a more nuanced position: these empirical results may not indicate that fine-tuning creates pragmatic abilities, but rather that it makes latent capabilities in base models more readily accessible. This reinterpretation gains credibility from several observations. Base models do exhibit pragmatic understanding that scales with size, and their performance improves substantially with in-context examples. Indeed, it is difficult to know for sure base models would not have achieved the same performance as their post-trained counterparts through more complex inference-time techniques. Moreover, it is hard to know when the techniques themselves give the model too much outside signal for us to claim the base model has the capability on its own. Consider the scenarios where a base model achieves equivalent task performance through few-shot prompting compared to an instruction-tuned model's zero-shot performance; did the few-shot labels confer novel abilities? What about the situation where random-label examples that merely clarify task format yield similar improvements, or where best-of-N sampling on base models succeeds without external supervision. Each scenario complicates our understanding of capability emergence, and with today's knowledge I would be more careful with claims that a post-training method is important for improving pragmatic understanding.

7.2.4 Scaling Compute Infinitely

Having discussed the nuances of the results presented in this thesis, here I combine its findings with insights from the broader community and instead present the most optimistic view of where the compute scaling paradigm can lead, acknowledging the speculative nature of such claims. While this thesis demonstrates the remarkable generalisations that current LLMs make from their training data, it ultimately leaves open the central question I set out to answer: whether models can truly generate novel knowledge.

Scaling laws predict continued gains as long as model size and training data increase proportionally. Intuitively, a model that perfectly predicts tokens in an infinite data stream would necessarily possess a causal understanding of the underlying distribution. While this is unrealistic, the findings in this thesis demonstrate qualitatively different patterns of generalisation at scale. Even at a small scale, the community discovered sudden perfect generalisation when overtraining on algorithmic tasks, a phenomenon known as 'grokking' [Pow+22]. This emerges because regularisation techniques like weight decay favour efficient generalising solutions over brute-force memorisation [Wan+24a; Liu+22]. Could LLMs be grokking the diverse implicit tasks within their training data? As training scales, memorisation becomes increasingly inefficient compared to true understanding. While grokking has only been observed in algorithmic domains with strict logical structure, the underlying principle that more compute favours generalisation over memorisation may extend more broadly.

Griffiths [Gri20] argues human intelligence presents differently from artificial intelligence because of bottlenecks, in time, computation, and communication. Humans evolved under pressure to generalise rapidly from limited data, while AI systems face fewer such constraints and consequently require vastly more data to achieve generalisation. Yet this difference in efficiency may be irrelevant if AI can ultimately generalise effectively. LLMs can process many times more information than any single human can, communicate across broader networks, and operate at superhuman speed. Combined with genuinely generalisable reasoning, could this computational advantage eventually yield novel discoveries? The answer to this question remains uncertain, hinging on our ability to generate sufficient data, but the trajectory of progress suggests the scaling paradigm's potential should not be underestimated.

7.3 Concrete Future Directions

Landing back with both feet firmly on earth, in this section I briefly outline a few concrete future directions following from each content chapter in this thesis.

Using training data attribution to understand LLM capabilities. Large-scale data attribution for LLMs as demonstrated in Chapter 3 has been attempted only once before [Gro+23], leaving numerous questions unexplored.

The general framework introduced in Chapter 3, defining tasks by sets of prompt-completion pairs and analysing how models learn these from training data, can be used to study many more interesting questions. Our investigation focuses on ranking pre-training data based on their influence on the likelihood of chain-of-thought traces with step-by-step arithmetic. Future work could examine whether influence rankings quantitatively and qualitatively differ when models output answers directly without chain-of-thought scaffolding, how semantically equivalent prompt variations affect these rankings, or whether different reasoning domains like social cognition exhibit distinct attribution patterns. Further, it opens up different directions, like finding evidence for the simulator hypothesis [And22; SMR23]. Do prompts asking for the same task completion, but written in very different ways (e.g. "you are a highly intelligent ..") lead to TDA methods surfacing higher quality data sources? Finally, how does few-shot prompting change rankings compared to zero-shot prompts?

Using training data attribution to understand emergence with scale.
Our comparison between 7B and 35B parameter models in Chapter 3 reveals striking quantitative differences in learning from data. Individual documents exert substantially larger influence on completion likelihood in the larger model, potentially indicating improved data efficiency. More puzzling, a document's influence on the 7B model's completion mostly bears no predictive relationship to its influence on the 35B model's response to identical prompts. Even when both models coincidentally generated identical completions to the same prompt, their training data influence scores show only a Pearson's R correlation of 0.19. This suggests fundamental differences in how models of different scales extract and utilise information from training data. Future research could identify systematic patterns distinguishing what data benefits smaller versus larger models, ultimately characterising how learning mechanisms evolve with scale.

Out-of-context probabilistic reasoning.
The generalisation from code generation to code evaluation we observe in Chapter 4 is a form of out-of-context reasoning: the ability to infer knowledge implicit within disparate pieces of training data and apply that knowledge downstream. In this case, the model combines familiar logical operations in an unseen configuration. An intriguing extension emerges when considering probabilistic rather than deterministic reasoning: can models develop out-of-context capabilities for frequentist or even Bayesian reasoning? For example,

imagine fine-tuning LLMs on datasets representing a certain probability distribution. Overall, in 80% of the data points wugs are red, and in 20% they are blue. Moreover, red wugs have a 50% chance to be peaceful, whereas blue wugs only have a 10% chance to be peaceful. Training on this data, can the LLM articulate the probability that a peaceful wug is red? More broadly, probabilistic and inductive out-of-context reasoning abilities could lead to interesting applications in scientific research, where training models on data from experiments may lead to their ability to articulate theories explaining this data.

Generalisation from verifiable reasoning to less easily verifiable reasoning.
For LLM reasoning to be broadly useful, it must extend beyond tasks with clear, automatically checkable answers to domains where correctness is harder or impossible to determine directly. Many of the recent advances in LLM reasoning have come from large-scale training on tasks with verifiable rewards (e.g. mathematics, code execution). A promising direction for future work is to use similar controlled fine-tuning experiments as in Chapter 4 to investigate whether models trained on verifiable reasoning problems can generalise to less verifiable settings, such as social reasoning, or to empirically verifiable domains like engineering in machine learning, where reasoning must be validated through experiments rather than ground-truth labels.

Comparing passive learning and active learning.
In Chapter 4, we find that on-policy reinforcement learning with verified rewards (RLVR) substantially outperforms passive approaches, whether training on expert demonstrations or off-policy learning from reward-model–labelled positive and negative samples. Therefore, we attribute these gains to the on-policy nature of the data used for training. On-policy training both results in a better fit to the current model distribution, as well as less constraints in generation. The model can generate more diverse completions for which it will receive rewards. Can future research demarcate between these two, pinpointing more clearly what about on-policy RLVR makes it more useful? Extending beyond this question, can models learn qualitatively different things from active *interaction* with an environment than when learning is done passively? This question is difficult to study because comparing these two head-to-head while only varying interaction is challenging, as demonstrated by animal studies

on the topic. For example, in Held and Hein [HH63b]'s "kitten carousel" experiment, two kittens were physically attached to the same enclosed carousel right after birth so that they received identical visual input, but only one could move and thereby control the other's ride, isolating the effect of self-guided interaction from mere exposure. Only the active kitten developed normal visuomotor coordination. Can we similarly design an experimental protocol to isolate the effect of interactive learning for LLMs?

How does pragmatic understanding emerge?

Chapter 5 demonstrates that while pragmatic understanding of language emerges during pre-training, it substantially improves through post-training. This raises a question about the nature of this improvement: does post-training genuinely teach new pragmatic reasoning capabilities, or does it primarily surface and amplify latent abilities already present from pre-training? We can disentangle these mechanisms through several approaches. First, inference-time methods like best-of-N sampling applied to base models could reveal the upper bounds of their pragmatic capabilities before post-training. Second, controlled post-training experiments could isolate the specific factors driving pragmatic improvements. For instance, if post-training exclusively on logical reasoning data such as MathInstruct [Tos+24] also enhances pragmatic understanding, this would suggest that post-training primarily serves to surface existing pragmatic understanding in base models.

Towards a developmental psychology of LLMs.

Chapter 6 presents inconclusive findings regarding whether LLMs encode text differently based on agent animacy: a crucial precursor to the pragmatic abilities demonstrated in Chapter 5. These results suggest that LLM social reasoning may not follow human developmental trajectories, raising fundamental questions about machine cognition. An exciting direction for future work involves essentially reconstructing developmental psychology for artificial systems: mapping which reasoning capabilities emerge at what scales and in what sequence across model development. This research program could extend beyond replicating human developmental milestones to discovering uniquely machine forms of social cognition. For instance, we might develop a machine theory of mind: a framework for understanding how LLMs represent and reason about mental states that may bear little resemblance to human theory of mind mechanisms. Such investigations could reveal whether LLMs develop alternative but equally sophisticated approaches to modelling agency, intentionality, and social reasoning, potentially uncovering computational strategies for social cognition that evolution never explored in biological systems.

Bibliography

[Aky+23] Ekin Akyürek et al. “What learning algorithm is in-context learning? Investigations with linear models”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=0g0X4H8yN4I.

[AL23] Zeyuan Allen-Zhu and Yuanzhi Li. “Physics of Language Models: Part 3.2, Knowledge Manipulation”. In: *arXiv* 2309.14402 (2023). URL: https://doi.org/10.48550/arXiv.2309.14402.

[And22] Jacob Andreas. “Language Models as Agent Models”. In: *ArXiv* abs/2212.01681 (2022).

[Ary+24] Viraat Aryabumi et al. “To Code, or Not To Code? Exploring Impact of Code in Pre-training”. In: *arXiv* 2408.10914 (2024). URL: https://arxiv.org/2408.10914.

[Bac99] Kent Bach. “The Myth of Conventional Implicature”. In: *Linguistics and Philosophy* 22.4 (1999), pp. 327–366. DOI: 10.1023/a:1005466020243.

[Bae+22] Juhan Bae et al. “If Influence Functions are the Answer, Then What is the Question?” In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=hzbguA9zMJ.

[Bae+24] Juhan Bae et al. *Training Data Attribution via Approximate Unrolled Differentiation*. 2024. arXiv: 2405.12186 [cs.LG]. URL: https://arxiv.org/abs/2405.12186.

[Bak+17] Chris L. Baker et al. “Rational quantitative attribution of beliefs, desires and percepts in human mentalizing”. In: *Nature Human Behaviour* 1 (2017).

[BBD20] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. "RelatIF: Identifying Explanatory Training Samples via Relative Influence". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, Aug. 2020, pp. 1899–1909. URL: https://proceedings.mlr.press/v108/barshan20a.html.

[BCB15] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *International Conference on Learning Representations*. 2015.

[Ben+03] Yoshua Bengio et al. "A neural probabilistic language model". In: *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435.

[Ben+21] Emily M Bender et al. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 2021, pp. 610–623.

[Ber+23] Lukas Berglund et al. "Taken out of context: On measuring situational awareness in LLMs". In: *arXiv* 2309.00667 (2023). URL: https://doi.org/10.48550/arXiv.2309.00667.

[Bet+25a] Jan Betley et al. "Emergent Misalignment: Narrow finetuning can produce broadly misaligned LLMs". In: *arXiv* 2502.17424 (2025). URL: https://arxiv.org/2502.17424.

[Bet+25b] Jan Betley et al. "Tell me about yourself: LLMs are aware of their learned behaviors". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=IjQ2Jtemzy.

[Big22] BigScience. *BigScience Language Open-science Open-access Multilingual (BLOOM) Language Model*. May 2022. URL: https://huggingface.co/bigscience/bloom.

[BK20] Emily M. Bender and Alexander Koller. "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*. Ed. by Dan Jurafsky et al. Online:

Association for Computational Linguistics, July 2020, pp. 5185–5198. DOI: 10.18653/v1/2020.acl-main.463. URL: https://aclanthology.org/2020.acl-main.463/.

[Bla+22] Sid Black et al. “GPT-NeoX-20B: An Open-Source Autoregressive Language Model”. In: *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*. 2022. URL: https://arxiv.org/abs/2204.06745.

[Boj+17] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5 (2017). Ed. by Lillian Lee, Mark Johnson, and Kristina Toutanova, pp. 135–146. DOI: 10.1162/tacl_a_00051. URL: https://aclanthology.org/Q17-1010/.

[BPF21] Samyadeep Basu, Phil Pope, and Soheil Feizi. “Influence Functions in Deep Learning Are Fragile”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=xHKVVHGDOEk.

[Bri+23] Trenton Bricken et al. “Towards Monosemanticity: Decomposing Language Models With Dictionary Learning”. In: *Transformer Circuits Thread* (2023). https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[Bro+20a] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[Bro+20b] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[BST09] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. “Action understanding as inverse planning”. In: *Cognition* 113 (2009), pp. 329–349.

[BVP18] Sara Beery, Grant Van Horn, and Pietro Perona. "Recognition in Terra Incognita". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Cha+22] Stephanie C.Y. Chan et al. "Data Distributional Properties Drive Emergent In-Context Learning in Transformers". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=lHj-q9BSRjF.

[Cho+22] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

[Cho+24] Sang Keun Choe et al. *What is Your Data Worth to GPT? LLM-Scale Data Valuation with Influence Functions*. 2024. arXiv: 2405.13954 [cs.LG]. URL: https://arxiv.org/abs/2405.13954.

[Chu+22] Hyung Won Chung et al. *Scaling Instruction-Finetuned Language Models*. 2022. DOI: 10.48550/ARXIV.2210.11416. URL: https://arxiv.org/abs/2210.11416.

[Chu+25] Tianzhe Chu et al. "SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training". In: *arXiv* 2501.17161 (2025). URL: https://doi.org/10.48550/arXiv.2501.17161.

[Cia+18] Andre Cianflone et al. "Let's do it "again": A First Computational Approach to Detecting Adverbial Presupposition Triggers". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2747–2755. DOI: 10.18653/v1/P18-1256. URL: https://aclanthology.org/P18-1256.

[Coh+25] Team Cohere et al. "Command A: An Enterprise-Ready Large Language Model". In: *arXiv* 2504.00698 (2025). URL: https://arxiv.org/2504.00698.

[Coo+26] Jonathan Cook et al. *Programming by Backprop: One Instruction is Worth One Hundred Examples When Training LLMs*. 2026. URL: https://openreview.net/forum?id=y1OWj26FCo.

[Das16] Dask Development Team. *Dask: Library for dynamic task scheduling*. 2016. URL: http://dask.pydata.org.

[Dav19] Wayne Davis. “Implicature”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2019. Metaphysics Research Lab, Stanford University, 2019. URL: https://plato.stanford.edu/archives/fall2019/entries/implicature/.

[Dav98] Wayne A. Davis. *Implicature : intention, convention, and principle in the failure of Gricean theory / Wayne A. Davis.* eng. Cambridge studies in philosophy. Cambridge England ; New York: Cambridge University Press, 1998. ISBN: 0521623197.

[dDD20] Baudouin Forgeot d’Arc, Marie Devaine, and Jean Daunizeau. “Social behavioural adaptation in Autism”. In: *PLoS Computational Biology* 16 (2020).

[Den+24] Chunyuan Deng et al. “Benchmark Probing: Investigating Data Leakage in Large Language Models”. In: *NeurIPS 2023 Workshop on Backdoors in Deep Learning - The Good, the Bad, and the Ugly*. 2024. URL: https://openreview.net/forum?id=a34bgvner1.

[Dev+17] Marie Devaine et al. “Reading wild minds: A computational assay of Theory of Mind sophistication across seven primate species”. In: *PLoS Computational Biology* 13 (2017).

[Dev+18] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[DHD14] Marie Devaine, Guillaume Hollard, and Jean Daunizeau. “The Social Bayesian Brain: Does Mentalizing Make a Difference When We Learn?” In: *PLoS Computational Biology* 10 (2014).

[Dua+19] Dheeru Dua et al. “DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs”. In: *Proc. of NAACL*. 2019.

[Dub+24] Abhimanyu Dubey et al. “The Llama 3 Herd of Models”. In: *arXiv* 2407.21783 (2024). DOI: 10.48550/ARXIV.2407.21783. URL: https://doi.org/10.48550/arXiv.2407.21783.

[EL20] Avia Efrat and Omer Levy. “The Turking Test: Can Language Models Understand Instructions?” In: *CoRR* abs/2010.11982 (2020). arXiv: 2010.11982. URL: https://arxiv.org/abs/2010.11982.

[Elh+22] Nelson Elhage et al. "Toy Models of Superposition". In: *Transformer Circuits Thread* (2022). https://transformer-circuits.pub/2022/toy_model/index.html.

[Elm90] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[Fan64] Robert L. Fantz. "Visual Experience in Infants: Decreased Attention to Familiar Patterns Relative to Novel Ones". In: *Science* 146.3644 (1964), pp. 668–670. DOI: 10.1126/science.146.3644.668. eprint: https://www.science.org/doi/pdf/10.1126/science.146.3644.668. URL: https://www.science.org/doi/abs/10.1126/science.146.3644.668.

[FF12] Chris D. Frith and Uta Frith. "Mechanisms of social cognition." In: *Annual review of psychology* 63 (2012), pp. 287–313.

[FG12] Michael C. Frank and Noah D. Goodman. "Predicting Pragmatic Reasoning in Language Games". In: *Science* 336 (2012), pp. 998–998.

[Fri17] Angela D. Friederici. *Language in Our Brain: The Origins of a Uniquely Human Capacity*. The MIT Press, Nov. 2017. ISBN: 9780262342964. DOI: 10.7551/mitpress/11173.001.0001. eprint: https://direct.mit.edu/book-pdf/2259328/book_9780262342964.pdf. URL: https://doi.org/10.7551/mitpress/11173.001.0001.

[FT25] FT. *Tech utterly dominates markets. Should we worry?* 2025. URL: https://www.ft.com/content/2a984f31-5c91-4301-91a6-8a62f3adc714.

[Gan+21] Kanishk Gandhi et al. "Baby Intuitions Benchmark (BIB): Discerning the goals, preferences, and actions of others". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=TFEFvU0ZV6Q.

[Gan+23] Kanishk Gandhi et al. "Understanding Social Reasoning in Language Models with Language Models". In: *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2023. URL: https://openreview.net/forum?id=8bqjirgxQM.

[Gan+24] Kanishk Gandhi et al. "Stream of Search (SoS): Learning to Search in Language". In: *arXiv* 2404.03683 (2024). URL: https://doi.org/10.48550/arXiv.2404.03683.

[Gao+21] Leo Gao et al. *A framework for few-shot language model evaluation*. Version v0.0.1. Sept. 2021. DOI: 10.5281/zenodo.5371628. URL: https://doi.org/10.5281/zenodo.5371628.

[GCT24] T.L. Griffiths, N. Chater, and J.B. Tenenbaum. *Bayesian Models of Cognition: Reverse Engineering the Mind*. MIT Press, 2024. ISBN: 9780262381048. URL: https://books.google.nl/books?id=FEb6EAAAQBAJ.

[Gem+25] Team Gemma et al. "Gemma 3 Technical Report". In: *arXiv* 2503.19786 (2025). URL: https://arxiv.org/2503.19786.

[Geo+18] Thomas George et al. "Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/48000647b315f6f00f913caa757a70b3-Paper.pdf.

[GF16] Noah Goodman and Michael Frank. "Pragmatic Language Interpretation as Probabilistic Inference". In: *Trends in Cognitive Sciences* 20 (Sept. 2016). DOI: 10.1016/j.tics.2016.08.005.

[Gla+22] Amelia Glaese et al. *Improving alignment of dialogue agents via targeted human judgements*. Sept. 2022. URL: https://storage.googleapis.com/deepmind-media/DeepMind.com/Authors-Notes/sparrow/sparrow-final.pdf.

[GM20] Elizabeth Jasmi George and Radhika Mamidi. "Conversational implicatures in English dialogue: Annotated dataset". In: *Procedia Computer Science* 171 (2020). https://doi.org/10.1016/j.procs.2020.04.251, pp. 2316–2323. DOI: 10.1016/j.procs.2020.04.251. URL:

https://app.dimensions.ai/details/publication/pub.1128198497.

[GR09] Stephan Greene and Philip Resnik. "More than Words: Syntactic Packaging and Implicit Sentiment". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado: Association for Computational Linguistics, June 2009, pp. 503–511. URL: https://aclanthology.org/N09-1057.

[Gre96] G.M. Green. *Pragmatics and Natural Language Understanding*. Tutorial essays in cognitive science. Erlbaum, 1996. ISBN: 9780805821659.

[Gri20] Thomas L. Griffiths. "Understanding Human Intelligence through Human Limitations". In: *Trends in Cognitive Sciences* 24.11 (2020), pp. 873–883. ISSN: 1364-6613. DOI: https://doi.org/10.1016/j.tics.2020.09.001. URL: https://www.sciencedirect.com/science/article/pii/S1364661320302151.

[Gri75] H. P. Grice. "Logic and Conversation". In: *Syntax and Semantics: Vol. 3: Speech Acts*. Ed. by Peter Cole and Jerry L. Morgan. New York: Academic Press, 1975, pp. 41–58. URL: http://www.ucl.ac.uk/ls/studypacks/Grice-Logic.pdf.

[Gro+23] Roger Grosse et al. *Studying Large Language Model Generalization with Influence Functions*. 2023. arXiv: 2308.03296 [cs.LG]. URL: https://arxiv.org/abs/2308.03296.

[Gun+23] Suriya Gunasekar et al. *Textbooks Are All You Need*. 2023. arXiv: 2306.11644 [cs.CL]. URL: https://arxiv.org/abs/2306.11644.

[Ham74] Frank R. Hampel. "The Influence Curve and its Role in Robust Estimation". In: *Journal of the American Statistical Association* 69.346 (1974), pp. 383–393. DOI: 10.1080/01621459.1974.10482962. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1974.10482962. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1974.10482962.

[HCF02] Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. "The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?" In: *Science* 298.5598 (2002), pp. 1569–1579. DOI: 10.1126/science.298.5598.1569. eprint: https://www.science.org/doi/pdf/10.1126/science.298.5598.1569. URL: https://www.science.org/doi/abs/10.1126/science.298.5598.1569.

[HFg] HF:greengerong. "leetcode". In: (). URL: https://huggingface.co/datasets/greengerong/leetcode.

[HH63a] Richard Held and Alan Hein. "Movement-Produced Stimulation in the Development of Visually Guided Behavior". In: *Journal of comparative and physiological psychology* 56 (Oct. 1963), pp. 872–6. DOI: 10.1037/h0040546.

[HH63b] Richard Held and Alan Hein. "Movement-Produced Stimulation in the Development of Visually Guided Behavior". In: *Journal of comparative and physiological psychology* 56 (Oct. 1963), pp. 872–6. DOI: 10.1037/h0040546.

[HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". In: *SIAM Review* 53.2 (2011), pp. 217–288. DOI: 10.1137/090771806. eprint: https://doi.org/10.1137/090771806. URL: https://doi.org/10.1137/090771806.

[Hof+22] Jordan Hoffmann et al. "An empirical analysis of compute-optimal large language model training". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., 2022, pp. 30016–30030. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

[Hos+23] Mohammad Javad Hosseini et al. "Resolving Indirect Referring Expressions for Entity Selection". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 12313–12335. DOI: 10.18653/v1/2023.

acl-long.688. URL: https://aclanthology.org/2023.acl-long.688.

[Hua17] Y. Huang. *The Oxford Handbook of Pragmatics*. Oxford handbooks in linguistics. Oxford University Press, 2017. ISBN: 9780199697960. URL: https://books.google.de/books?id=PlvjDQAAQBAJ.

[Jai+24] Samyak Jain et al. "Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=A0HKeKl4Nl.

[Jer+20] Paloma Jeretic et al. "Are Natural Language Inference Models IMPPRESsive? Learning IMPlicature and PRESupposition". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8690–8705. DOI: 10.18653/v1/2020.acl-main.768. URL: https://aclanthology.org/2020.acl-main.768.

[Kah03] Daniel Kahneman. "A Perspective on Judgment and Choice: Mapping Bounded Rationality". In: *American Psychologist* 58.9 (2003), p. 697. DOI: 10.1037/0003-.

[Kap+20] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

[KB15] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA, 2015.

[Kim+21] Najoung Kim et al. "Which Linguist Invented the Lightbulb? Presupposition Verification for Question-Answering". 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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 3932–3945. DOI: 10.18653/v1/2021.acl-long.304. URL: https://aclanthology.org/2021.acl-long.304.

[Kir+24] Robert Kirk et al. "Understanding the Effects of RLHF on LLM Generalisation and Diversity". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. 2024. URL: https://openreview.net/forum?id=PXD3FAVHJT.

[KL17] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1885–1894.

[Koj+22] Takeshi Kojima et al. *Large Language Models are Zero-Shot Reasoners*. 2022. arXiv: 2205.11916 [cs.CL].

[Kos24] Michal Kosinski. "Evaluating large language models in theory of mind tasks". In: *Proceedings of the National Academy of Sciences* 121.45 (2024), e2405460121. DOI: 10.1073/pnas.2405460121. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2405460121. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2405460121.

[Kra+24] Dmitrii Krasheninnikov et al. "Implicit meta-learning may lead language models to trust more reliable sources". In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=Fzp1DRzCIN.

[KSR24] Suhas Kotha, Jacob Mitchell Springer, and Aditi Raghunathan. "Understanding Catastrophic Forgetting in Language Models via Implicit Inference". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=VrHiF2hsrm.

[Lai+17] Guokun Lai et al. "RACE: Large-scale ReAding Comprehension Dataset From Examinations". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 785–794. DOI: 10.18653/v1/D17-1082. URL: https://aclanthology.org/D17-1082.

[Lam+25] Andrew K. Lampinen et al. "On the generalization of language models from in-context learning and finetuning: a controlled study". In: *arXiv* 2505.00661 (2025). URL: https://arxiv.org/2505.00661.

[Lan00] P. Langley. "Crafting Papers on Machine Learning". In: *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*. Ed. by Pat Langley. Stanford, CA: Morgan Kaufmann, 2000, pp. 1207–1216.

[Las+23] Michael Laskin et al. "In-context Reinforcement Learning with Algorithm Distillation". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/forum?id=hy0a5MMPUv.

[LB18] Brenden Lake and Marco Baroni. "Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks". English (US). In: *35th International Conference on Machine Learning, ICML 2018*. Ed. by Jennifer Dy and Andreas Krause. 35th International Conference on Machine Learning, ICML 2018. Publisher Copyright: © Copyright 2018 by the author(s).; 35th International Conference on Machine Learning, ICML 2018 ; Conference date: 10-07-2018 Through 15-07-2018. International Machine Learning Society (IMLS), 2018, pp. 4487–4499.

[Lev83] Stephen C. Levinson. *Pragmatics*. Cambridge, U.K.: Cambridge University Press, 1983.

[Liu+19] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[Liu+22] Ziming Liu et al. "Towards Understanding Grokking: An Effective Theory of Representation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=6at6rB3IZm.

[LPM15] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods* *in Natural Language Processing*. Ed. by Lluís Màrquez, Chris Callison-Burch, and Jian Su. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166/.

[LRR20] Annie Louis, Dan Roth, and Filip Radlinski. ""I'd rather just go to bed": Understanding Indirect Answers". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 7411–7425. DOI: 10.18653/v1/2020.emnlp-main.601. URL: https://aclanthology.org/2020.emnlp-main.601.

[LS14] Ernie Lepore and Matthew Stone. *Imagination and Convention: Distinguishing Grammar and Inference in Language*. Oxford University Press, Dec. 2014. ISBN: 9780198717188. DOI: 10.1093/acprof:oso/9780198717188.001.0001. URL: https://doi.org/10.1093/acprof:oso/9780198717188.001.0001.

[LSD21] Elissa Li, Sebastian Schuster, and Judith Degen. "Predicting scalar inferences from "or" to "not both" using neural sentence encoders". In: *Proceedings of the Society for Computation in Linguistics 2021*. Online: Association for Computational Linguistics, Feb. 2021, pp. 446–450. URL: https://aclanthology.org/2021.scil-1.54.

[Lu+22] Yao Lu et al. "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity". In: *ACL*. 2022.

[MAD07] Karen Milligan, Janet Wilde Astington, and Lisa Ain Dack. "Language and Theory of Mind: Meta-Analysis of the Relation Between Language Ability and False-belief Understanding". In: *Child Development* 78.2 (2007), pp. 622–646. DOI: https://doi.org/10.1111/j.1467-8624.2007.01018.x. eprint: https://srcd.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8624.2007.01018.x. URL: https://srcd.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8624.2007.01018.x.

[Mah+24] Kyle Mahowald et al. "Dissociating language and thought in large language models". In: *Trends in Cognitive Sciences* 28 (Mar. 2024). DOI: 10.1016/j.tics.2024.01.011.

[McC+23] R. Thomas McCoy et al. *Embers of Autoregression: Understanding Large Language Models Through the Problem They are Trained to Solve*. 2023. arXiv: 2309.13638 [cs.CL]. URL: https://arxiv.org/abs/2309.13638.

[McC+24] R. Thomas McCoy et al. "Embers of autoregression show how large language models are shaped by the problem they are trained to solve". In: *Proceedings of the National Academy of Sciences* https://www.pnas.org/doi/pdf/10.1073/pnas.2322420121.41 (2024), e2322420121. DOI: 10.1073/pnas.2322420121. URL: https://www.pnas.org/doi/10.1073/pnas.2322420121.

[McL+24] Sean McLeish et al. *Transformers Can Do Arithmetic with the Right Embeddings*. 2024. arXiv: 2405.17399 [cs.LG]. URL: https://arxiv.org/abs/2405.17399.

[Mer+16] Stephen Merity et al. *Pointer Sentinel Mixture Models*. 2016. arXiv: 1609.07843 [cs.CL].

[MG15] James Martens and Roger Grosse. "Optimizing neural networks with Kronecker-factored approximate curvature". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 2408–2417.

[MH23] Sh.R. Moghaddam and Christopher John Honey. "Boosting Theory-of-Mind Performance in Large Language Models via Prompting". In: *ArXiv* abs/2304.11490 (2023).

[MS18] Hugo Mercier and Dan Sperber. *The Enigma of Reason*. Penguin, 2018.

[MWL22] William Merrill, Alex Warstadt, and Tal Linzen. "Entailment Semantics Can Be Extracted from an Ideal Language Model". In: *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 176–193. URL: https://aclanthology.org/2022.conll-1.13.

[Ng+19] Nathan Ng et al. "Facebook FAIR's WMT19 News Translation Task Submission". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 314–319. DOI: 10.18653/v1/W19-5333. URL: https://aclanthology.org/W19-5333.

[Nik+25] Yaniv Nikankin et al. "Arithmetic Without Algorithms: Language Models Solve Math with a Bag of Heuristics". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=O9YTt26r2P.

[Nov+25] Alexander Novikov et al. "AlphaEvolve: A coding agent for scientific and algorithmic discovery". In: *arXiv* 2506.13131 (2025). URL: https://arxiv.org/2506.13131.

[NYT25] NYT. *Is the A.I. Sell-Off the Start of Something Bigger?* 2025. URL: https://www.nytimes.com/2025/08/20/business/dealbook/ai-dip-blip-palantir-nvidia.html.

[OCD21] Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. "The Difficulty of Passive Learning in Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=nPHA8fGicZk.

[Ols+22] Catherine Olsson et al. "In-context Learning and Induction Heads". In: *Transformer Circuits Thread* (2022). https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[Ouy+22] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *ArXiv* abs/2203.02155 (2022).

[Par+21] Alicia Parrish et al. "NOPE: A Corpus of Naturally-Occurring Presuppositions in English". In: *Proceedings of the 25th Conference on Computational Natural Language Learning*. Online: Association for Computational Linguistics, Nov. 2021, pp. 349–366. DOI: 10.18653/v1/2021.conll-1.28. URL: https://aclanthology.org/2021.conll-1.28.

[PJ05] Steven Pinker and Ray Jackendoff. "The faculty of language: what's special about it?" In: *Cognition* 95.2 (2005), pp. 201–236. ISSN: 0010-0277. DOI: https://doi.org/10.1016/j.cognition.2004.08.004. URL: https://www.sciencedirect.com/science/article/pii/S0010027704001763.

[Pot05] Christopher Potts. *The Logic of Conventional Implicatures*. Oxford University Press UK, 2005.

[Pot06] Christopher Potts. "Conversational Implicatures Via General Pragmatic Pressures". In: *New Frontiers in Artificial Intelligence*. Ed. by Takashi Washio et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 205–218. ISBN: 978-3-540-69902-6.

[Pow+22] Alethea Power et al. "Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets". In: *CoRR* abs/2201.02177 (2022). arXiv: 2201.02177. URL: https://arxiv.org/abs/2201.02177.

[PP21] Roma Patel and Ellie Pavlick. ""Was it "stated" or was it "claimed"?: How linguistic bias affects generative language models". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10080–10095. DOI: 10.18653/v1/2021.emnlp-main.790. URL: https://aclanthology.org/2021.emnlp-main.790.

[PPF25] Ulyana Piterbarg, Lerrel Pinto, and Rob Fergus. "Training Language Models on Synthetic Edit Sequences Improves Code Synthesis". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=AqfUa08PCH.

[Pra+24] Nikhil Prakash et al. "Fine-Tuning Enhances Existing Mechanisms: A Case Study on Entity Tracking". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=8sKcAWOf2D.

[Pru+20] Garima Pruthi et al. "Estimating Training Data Influence by Tracing Gradient Descent". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19920–19930. URL: https:

//proceedings.neurips.cc/paper_files/paper/2020/file/e6385d39ec9394f2f3a354d9d2b88eec-Paper.pdf.

[PSL24] Jackson Petty, Sjoerd van Steenkiste, and Tal Linzen. "How Does Code Pretraining Affect Language Model Task Performance?" In: *arXiv* 2409.04556 (2024). URL: https://doi.org/10.48550/arXiv.2409.04556.

[Qwe+25] Qwen et al. "Qwen2.5 Technical Report". In: *arXiv* 2412.15115 (2025). URL: https://arxiv.org/2412.15115.

[Rad+18] Alec Radford et al. *Improving language understanding by generative pre-training*. 2018.

[Rad+19] Alec Radford et al. *Language Models are Unsupervised Multitask Learners*. 2019.

[Raf+23] Rafael Rafailov et al. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=HPuSIXJaa9.

[Raz+22] Yasaman Razeghi et al. "Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 840–854. DOI: 10.18653/v1/2022.findings-emnlp.59. URL: https://aclanthology.org/2022.findings-emnlp.59.

[RDJ13] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. "Linguistic Models for Analyzing and Detecting Biased Language". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 1650–1659. URL: https://aclanthology.org/P13-1162.

[RM21a] Laria Reynolds and Kyle McDonell. *Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm*. 2021. arXiv: 2102.07350 [cs.CL].

[RM21b] Laria Reynolds and Kyle McDonell. "Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm". In: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI EA '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380959. DOI: 10.1145/3411763.3451760. URL: https://doi.org/10.1145/3411763.3451760.

[Rui+23a] Laura Ruis et al. "Do LLMs selectively encode the goal of an agent's reach?" In: *First Workshop on Theory of Mind in Communicating Agents*. 2023. URL: https://openreview.net/forum?id=KxvXjtyuYl.

[Rui+23b] Laura Eline Ruis et al. "The Goldilocks of Pragmatic Understanding: Fine-Tuning Strategy Matters for Implicature Resolution by LLMs". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=5bWW9Eop7l.

[Rui+25] Laura Ruis et al. "Procedural Knowledge in Pretraining Drives Reasoning in Large Language Models". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=1hQKHHUsMx.

[San+22] Victor Sanh et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=9Vrb9D0WI4.

[Sap+22] Maarten Sap et al. "Neural Theory-of-Mind? On the Limits of Social Intelligence in Large LMs". In: *Conference on Empirical Methods in Natural Language Processing*. 2022.

[SCD20] Sebastian Schuster, Yuxing Chen, and Judith Degen. "Harnessing the linguistic signal to predict scalar inferences". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5387–5403. DOI: 10.18653/v1/2020.acl-main.479. URL: https://aclanthology.org/2020.acl-main.479.

[Sch+17] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[SGG14] Patrick Shafto, Noah D. Goodman, and Thomas L. Griffiths. "A rational account of pedagogical reasoning: Teaching by, and learning from, examples". In: *Cognitive Psychology* 71 (2014), pp. 55–89.

[Sha+23] Natalie Shapira et al. *Clever Hans or Neural Theory of Mind? Stress Testing Social Reasoning in Large Language Models*. 2023. arXiv: 2305.14763 [cs.CL].

[Sha+24] Zhihong Shao et al. "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models". In: *arXiv* 2402.03300 (2024). URL: https://arxiv.org/2402.03300.

[Sho+25] Parshin Shojaee et al. "The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity". In: *arXiv* 2506.06941 (2025). URL: https://arxiv.org/2506.06941.

[Sin+24] Aaditya K Singh et al. "What needs to go right for an induction head? A mechanistic study of in-context learning circuits and their formation". In: *Forty-first International Conference on Machine Learning*. 2024. URL: https://openreview.net/forum?id=O8rrXl71D5.

[SMR23] Murray Shanahan, Kyle McDonell, and Laria Reynolds. "Role play with large language models". In: *Nature* 623.7987 (Nov. 2023), pp. 493–498. DOI: 10.1038/s41586-023-06647-8. URL: https://ideas.repec.org/a/nat/nature/v623y2023i7987d10.1038_s41586-023-06647-8.html.

[Sri+22] Aarohi Srivastava et al. *Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models*. 2022. DOI: 10.48550/ARXIV.2206.04615. URL: https://arxiv.org/abs/2206.04615.

[SW86] D. Sperber and D. Wilson. *Relevance: Communication and Cognition*. Language and thought series. Harvard University Press, 1986. ISBN: 9780674754768. URL: https://books.google.ca/books?id=1LkkAQAAMAAJ.

[Tan+21] Zhixing Tan et al. "Msp: Multi-stage prompting for making pre-trained language models better translators". In: *arXiv preprint arXiv:2110.06609* (2021).

[TCL07] Michael Tomasello, Malinda Carpenter, and Ulf Liszkowski. "A New Look at Infant Pointing". In: *Child development* 78 (May 2007), pp. 705–22. DOI: 10.1111/j.1467-8624.2007.01025.x.

[Tem+24] Adly Templeton et al. "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet". In: *Transformer Circuits Thread* (2024). URL: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

[TER19] HERBERT S. TERRACE. *Why Chimpanzees Can't Learn Language and Only Humans Can*. Columbia University Press, 2019. ISBN: 9780231171106. URL: http://www.jstor.org/stable/10.7312/terr17110 (visited on 08/06/2025).

[Tes+21] Stefano Teso et al. "Interactive Label Cleaning with Example-based Explanations". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=T6m9bNI7C__.

[Tho+24] Jonathan Thomm et al. "Limits of Transformer Language Models on Learning to Compose Algorithms". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/0e797d5139ad94fc2dc2080c09119f29-Abstract-Conference.html.

[Tom99] Michael Tomasello. *The Cultural Origins of Human Cognition*. Harvard University Press, 1999. ISBN: 9780674000704. URL: http://www.jstor.org/stable/j.ctvjsf4jc (visited on 08/01/2025).

[Tos+24] Shubham Toshniwal et al. "OpenMathInstruct-1: A 1.8 Million Math Instruction Tuning Dataset". In: *arXiv* 2402.10176 (2024). URL: https://arxiv.org/2402.10176.

[Tou+23] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[Tre+24] Johannes Treutlein et al. "Connecting the Dots: LLMs can Infer and Verbalize Latent Structure from Disparate Training Data". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons et al. 2024. URL: http://papers.nips.cc/paper%5C_files/paper/2024/hash/fe489a28a54583ee802b8e2955c024c2-Abstract-Conference.html.

[Tro+22] Sean Trott et al. "Do Large Language Models know what humans know?" In: *arXiv preprint arXiv:2209.01515* (2022).

[Ull23a] Tomer Ullman. *Large Language Models Fail on Trivial Alterations to Theory-of-Mind Tasks*. 2023. arXiv: 2302.08399 [cs.AI]. URL: https://arxiv.org/abs/2302.08399.

[Ull23b] Tomer David Ullman. "Large Language Models Fail on Trivial Alterations to Theory-of-Mind Tasks". In: *ArXiv* abs/2302.08399 (2023).

[Von+23] Johannes Von Oswald et al. "Transformers Learn In-Context by Gradient Descent". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 35151–35174. URL: https://proceedings.mlr.press/v202/von-oswald23a.html.

[VYG78] L. S. VYGOTSKY. *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, 1978. ISBN: 9780674576285. URL: http://www.jstor.org/stable/j.ctvjf9vz4 (visited on 08/01/2025).

[Wan+24a] Boshi Wang et al. *Grokked Transformers are Implicit Reasoners: A Mechanistic Journey to the Edge of Generalization*. 2024. arXiv: 2405.15071 [cs.CL]. URL: https://arxiv.org/abs/2405.15071.

[Wan+24b] Boshi Wang et al. "Grokked Transformers are Implicit Reasoners: A Mechanistic Journey to the Edge of Generalization". In: *arXiv* 2405.15071 (2024). URL: https://doi.org/10.48550/arXiv.2405.15071.

[WE74] P.C. Wason and J.ST.B.T. Evans. "Dual processes in reasoning?" In: *Cognition* 3.2 (1974), pp. 141–154. ISSN: 0010-0277. DOI: https://doi.org/10.1016/0010-0277(74)90017-1. URL: https://www.sciencedirect.com/science/article/pii/0010027774900171.

[Wei+22a] Jason Wei et al. *Chain of Thought Prompting Elicits Reasoning in Large Language Models*. 2022. DOI: 10.48550/ARXIV.2201.11903. URL: https://arxiv.org/abs/2201.11903.

[Wei+22b] Jason Wei et al. "Chain of Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=_VjQlMeSB_J.

[Wel90] Henry M. Wellman. *The Child's Theory of Mind*. MIT Press (MA), 1990.

[WGM25] Yiwei Wu, Atticus Geiger, and Raphaël Millière. "How Do Transformers Learn Variable Binding in Symbolic Programs?" In: *Forty-Second International Conference on Machine Learning*. 2025.

[WHL23] Taylor Webb, Keith Holyoak, and Hongjing Lu. "Emergent analogical reasoning in large language models". In: *Nature Human Behaviour* 7 (July 2023), pp. 1–16. DOI: 10.1038/s41562-023-01659-w.

[WHL24] Taylor Webb, Keith J. Holyoak, and Hongjing Lu. *Evidence from counterfactual tasks supports emergent analogical reasoning in large language models*. 2024. arXiv: 2404.13070 [cs.CL]. URL: https://arxiv.org/abs/2404.13070.

[Wit21] L. Wittgenstein. "Tractatus Logico-Philosophicus". In: *London: Routledge, 1981* (1921). Ed. by D.F.Pears. URL: http://scholar.google.de/scholar.bib?q=info:1G2GoIkyCZIJ:scholar.google.com/&output=citation&hl=de&ct=citation&cd=0.

[Wit53] Ludwig Wittgenstein. *Philosophical Investigations*. Oxford: Basil Blackwell, 1953. ISBN: 0631119000.

[WK21] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[Wol+20] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6.

[Woo98] A. Woodward. "Infants Selectively Encode the Goal Object of an Actor's Reach". In: *Cognition* 69.1 (1998), pp. 1–34. DOI: 10.1016/s0010-0277(98)00058-4.

[WP21] Albert Webson and Ellie Pavlick. "Do Prompt-Based Models Really Understand the Meaning of their Prompts?" In: *arXiv preprint arXiv:2109.01247* (2021).

[WP83] Heinz Wimmer and Josef Perner. "Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception". In: *Cognition* 13.1 (1983), pp. 103–128. ISSN: 0010-0277. DOI: https://doi.org/10.1016/0010-0277(83)90004-5. URL: https://www.sciencedirect.com/science/article/pii/0010027783900045.

[Wu+24] Zhaofeng Wu et al. "Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Kevin Duh, Helena Gomez, and Steven Bethard. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 1819–1862. DOI: 10.18653/v1/2024.naacl-long.102. URL: https://aclanthology.org/2024.naacl-long.102.

[Wur+25] Daniel Wurgaft et al. *In-Context Learning Strategies Emerge Rationally*. 2025. arXiv: 2506.17859 [cs.LG]. URL: https://arxiv.org/abs/2506.17859.

[Yan+23] Shuo Yang et al. *Rethinking Benchmark and Contamination for Language Models with Rephrased Samples*. 2023. arXiv: 2311.04850 [cs.CL]. URL: https://arxiv.org/abs/2311.04850.

[YDF08] Wako Yoshida, Raymond J. Dolan, and Karl J. Friston. "Game Theory of Mind". In: *PLoS Computational Biology* 4 (2008).

[ZFM25] Zhenghao Zhou, Robert Frank, and R. Thomas McCoy. "Is In-Context Learning a Type of Error-Driven Learning? Evidence from the Inverse Frequency Effect in Structural Priming". In: *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Ed. by Luis Chiruzzo, Alan Ritter, and Lu Wang. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 11712–11725. ISBN: 979-8-89176-189-6. DOI: 10.18653/v1/2025.naacl-long.586. URL: https://aclanthology.org/2025.naacl-long.586/.

[Zha+22] Susan Zhang et al. *OPT: Open Pre-trained Transformer Language Models*. 2022. arXiv: 2205.01068 [cs.CL].

[Zhe+21] Zilong Zheng et al. "GRICE: A Grammar-based Dataset for Recovering Implicature and Conversational rEasoning". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, Aug. 2021, pp. 2074–2085. DOI: 10.18653/v1/2021.findings-acl.182. URL: https://aclanthology.org/2021.findings-acl.182.

[ZS15] Wojciech Zaremba and Ilya Sutskever. "Learning to Execute". In: *arXiv* 1410.4615 (2015). URL: https://arxiv.org/1410.4615.

Appendix A

How to Kick Your Appendix-Skipping Habit

Whoever reads the appendix, right? But this time, you might want to break that habit. Why? Because tucked away here are five sections that I think are genuinely worth your attention.

Starting with Section B.9.1.
In Chapter 3, I distinguished LLM reasoning from factual retrieval using correlation analysis: the influence of training documents on one reasoning question strongly predicts their influence on other reasoning questions of the same type. This suggests models generalise across questions, rather than just memorising. But could this simply be driven by superficial similarity between reasoning questions? Section B.9.1 shows that's not the case, and also offers a window into what models actually absorb from their pre-training data.

Other interesting sections.
Beyond that, there are four shorter contributions that don't sit at the centre of the thesis, but which I find interesting in their own right:

- **Section B.1:** EKFAC influence scores reveal how including a training document affects downstream reasoning accuracy.
- **Section B.8.2:** LLMs learn to answer English factual questions from documents containing the relevant facts in other languages.
- **Section C.4:** Online reinforcement learning is important for generalisation, as shown in compute-matched experiments comparing GRPO, DPO, and SFT.

- **Section D.7.6:** Randomising labels for few-shot in-context examples does not hurt performance, meaning they mainly serve to clarify task format instead of leaking information about the task.

Structure of the appendix.
I've designed the appendix to be maximally skippable: every section is flagged in the main text, and all key results are already summarised there. Still, if you're curious, each appendix chapter is about one main content chapter, with a short roadmap summarising contents at the top.

Appendix B

How Models Learn to Reason from Pre-training Data

This chapter contains the Appendix for Chapter 3. Below, I outline the content of each section in this appendix.

EKFAC influence functions. In Appendix B.1 we discuss the counterfactual re-training experiments that motivate our use of EKFAC influence functions for estimating the effect of pre-training data on the accuracy of downstream behaviour. We describe in more detail how we use influence functions at scale in Appendix B.2, documenting how we estimate the Hessian, how we store many query gradients in memory (each having the same memory complexity as the entire model), and how we sample from the pre-training distribution.

Query sets examples. Then, in Appendix B.3, we show examples of the reasoning sets that we did not show examples for in the main body of this manuscript.

Finding query answers in documents and characterising document-query relations. In Appendix B.4 we discuss how we create keywords for each query in order to find the answer in the top documents, and in the sections directly after that, Appendix B.5 and B.6, we give the prompts we used to allow Command R+ to search for answers in the top 500 documents for each query, as well as characterise their relationship.

Limitations. In Appendix B.7 we discuss limitations specific to influence functions.

Additional qualitative results. In Appendix B.8 we provide additional qualitative results.

Answer finding. We show examples of answer documents in Appendix B.8.1.

Cross-lingual transfer. We give some examples of cross-lingual transfer in

Appendix B.8.2.

Characterise query-document relation. We give detailed results on the characterisation of the relationship between queries and the top 500 documents in Appendix B.8.3.

Source-dataset analysis. We analyse which datasets the influential data comes from in Appendix B.8.4.

Content analysis of relevant documents. We classify data from the source dataset code for whether it actually contains code in Appendix B.8.5.

Additional quantitative results. In Appendix B.9 we provide additional quantitative results.

Correlation analysis. Further results for the correlation analysis of influence scores for documents for different queries in Appendix B.9.1.

Magnitude of influence. Further results for the magnitude of influence in Appendix B.9.2.

Spread of influence. Further results for the spread of influence over the rankings in Appendix B.9.3.

B.1 Counterfactual Re-training Experiments with Influence Functions

We use EKFAC influence functions to approximate the counterfactual question: which documents from pre-training have a causal effect on the completions of a trained model. However, we are also interested in the causal effect on the *accuracy* of the completions. In this section, we aim to motivate two aspects of this choice; the fact that influence functions are designed to estimate the effect on continuous differentiable functions, like the log-likelihood, and not on the accuracy. Secondly, we motivate the need for estimating the second-order information of the pre-training objective using EKFAC, which is very computationally expensive. We present four different experiments in this section, which show that indeed the influence of documents as determined by influence functions also estimate the effect on downstream task accuracy, as well as the benefits from estimating second order information over simply using first-order gradient information.

The pipeline for each of these experiments is similar; we take a pre-trained model, we fine-tune it on some dataset, and evaluate it on 50 validation examples with a metric (perplexity or accuracy). We then use the fine-tuned weights to calculate the influence of the documents in the dataset used for fine-tuning on the set of 50 validation questions with two methods: EKFAC influence functions and TracIn [Pru+20]. Subsequently, we use those two methods to remove the k most positively influential documents from the fine-tuning dataset, as well as randomly selecting k documents as a baseline, and fine-tune the original pre-trained model five times (with different seeds) on each new fine-tuning dataset created (for different values for k). We then calculate the perplexity or accuracy on the validation questions used to calculate the influence, and see how it changed. The more it changed, the more the documents indeed influence the relevant metric (i.e. perplexity or accuracy). Note that for n different values for k, this requires fine-tuning $3 * 5 * n$ models: five times for each of the three methods of removing documents from the training set.

We start by motivating the use of EKFAC influence functions over simple similarity information between document and query gradients. In our setup, where we only have access to the final checkpoint of pre-training, a dot-product between the query and document gradient effectively boils down to a method for estimating influence of documents on queries called TracIn [Pru+20]. With access to multiple checkpoints, TracIn uses gradient information from all of them, accounting for the learning rate used at that point in training. However, we only use the final checkpoint and hence taking into account learning rate only changes scores by a constant. We take GPT-2-small (124M) from HuggingFace,[1] and fine-tune it for three epochs with next-token prediction on Wikitext-2 [Mer+16]. We use Adam optimizer [KB15] with default parameters (b1 0.9, b2 0.999, eps 1e-8, additive weight decay 0.01). The results can be found in Figure B.1 and Table B.1, showing that removing documents using EKFAC influence functions has a significantly larger effect on downstream perplexity for all values of k. We do the exact same experiment but instead remove the most negatively influential documents, and see that instead the perplexity decreases significantly more for EKFAC influence functions (Figure B.1 and Table B.2).

Table B.1: Wikitext remove top influential

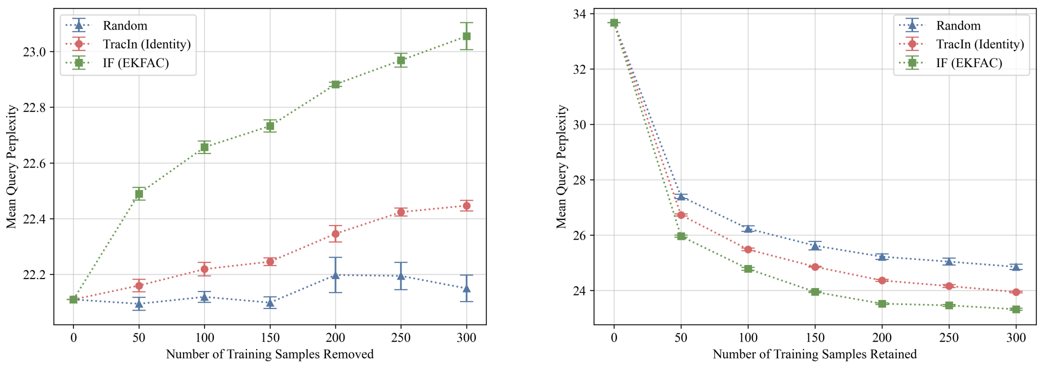
k →	50	100	150	200	250	300
Random	22.09 ± 0.02	22.12 ± 0.02	22.10 ± 0.02	22.20 ± 0.06	22.19 ± 0.05	22.15 ± 0.05
TracIn	22.16 ± 0.02**	22.22 ± 0.02**	22.25 ± 0.01**	22.35 ± 0.03**	22.42 ± 0.01**	22.45 ± 0.02**
IF (ours)	22.49 ± 0.02**	22.66 ± 0.02**	22.73 ± 0.02**	22.88 ± 0.01**	22.97 ± 0.02**	23.05 ± 0.05**

Table B.2: Wikitext remove bottom influential

k →	50	100	150	200	250	300
Random	27.40 ± 0.08	26.24 ± 0.10	25.62 ± 0.15	25.22 ± 0.10	25.04 ± 0.12	24.85 ± 0.10
TracIn	26.73 ± 0.04**	25.48 ± 0.05**	24.86 ± 0.02**	24.36 ± 0.04**	24.16 ± 0.05**	23.94 ± 0.03**
IF (ours)	25.96 ± 0.04**	24.78 ± 0.05**	23.95 ± 0.03**	23.52 ± 0.03**	23.46 ± 0.03**	23.32 ± 0.04**

Next, we turn to motivating the use of EKFAC influence functions in estimating the effect of documents on downstream accuracy of model generations. To this end, we look at two different datasets: DROP [Dua+19] and RACE [Lai+17]. DROP is a reading comprehension dataset requiring different skills like subtraction, addition, coreference resolution, counting, and other skills. The model needs to generate an answer that often consists of one or a few words. We allow the fine-tuned models to generate answers to the questions freely, and evaluate based on exact match. In this experiment, we use a 7B model. We randomly select a subset of 8000 examples for fine-tuning, and use the procedure described above to perform counterfactual experiments. We

[1] https://huggingface.co/



(a) **(b)**

Figure B.1: (a) Counterfactual retraining experiments on Wikitext-2. We fine-tuned GPT-2 (124M) on Wikitext-2 and use three different methods to remove training examples from the training set: randomly, TracIn, and Influence Functions (IF). For each number of samples removed we fine-tune the base model five times with different training data ordering, the variance over these runs is represented by the error bars. Each point on the plot is the average perplexity achieved by the five models after fine-tuning on the augmented dataset. We find that influence functions can find examples that impact the perplexity significantly more than baselines. (b) We repeat the same experiment as in (a), but retain top influential queries instead (removing most negatively influential).

use Adam optimizer again, with the same hyperparameters as for the above experiment: b1 0.9, b2 0.999, eps 1e-8, additive weight decay 0.01, but only train for one epoch. The results can be found in the left panel of Figure B.2 as well as in Table B.3. We find that EKFAC influence functions are succesful in selecting data points that impact downstream accuracy, much more so than randomly removing the same amount of training data. For most k (all but $k = 1000$), EKFAC influence functions also have a significantly stronger effect on accuracy than TracIn, but the difference is less large. We apply the exact same procedure to the RACE dataset, except now we keep 10k examples (empirically found to lead to the least overfitting when fine-tuning). Further, RACE is a multiple-choice dataset, so we allow the model to generate a single token indicating the choice, and calculate the accuracy. The results can be seen in Figure B.2 and Table B.4. Again, the finding is similar; EKFAC influence functions surface documents that have a stronger effect on accuracy than TracIn for all but one value of k, and for all values of k than randomly removing documents. There is a large variance in the results for all methods though, which we attribute to the fact that the model sometimes seems to overfit to the fine-tuning data. Further, the reason why the difference between TracIn and EKFAC influence functions is much larger in the perplexity experiments than in the accuracy experiments could be attributed to the fact that we only fine-tune for one epoch in the accuracy experiments (as more cause overfitting). EKFAC influence functions differ from TracIn in that they estimate second order information, which becomes more important with more training steps. An interesting avenue for future work is to do counterfactual re-training experiments like these on a subset of pre-training data for a 7B model, but this is incredibly computationally expensive.

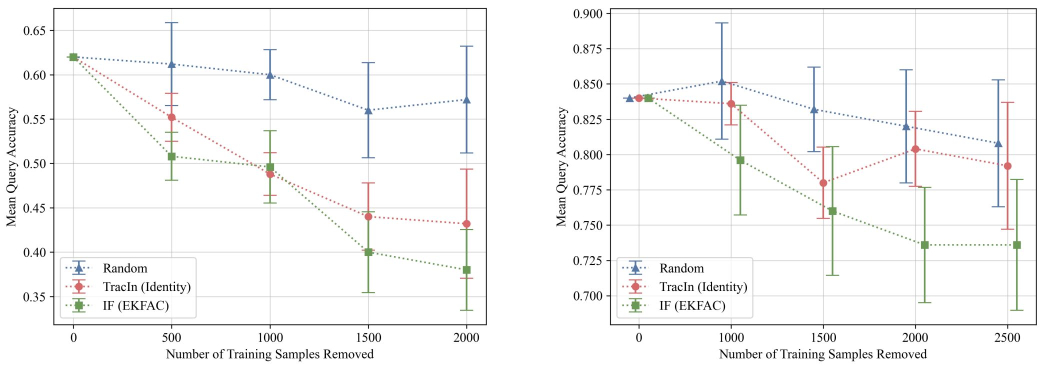
Table B.3: Counterfactual re-training accuracies on DROP (free generation of answers). We use three different methods (random, TracIn, influence functions) to remove k datapoints, and re-train a model on the resulting dataset. Each number is the mean over five re-training runs with different data ordering. $\star$ indicates significantly lower than random with a p-value below 0.1 and $\star\star$ with a p-value below 0.05. The underlined means are the lowest.

k $\rightarrow$	500	1000	1500	2000
Random	0.61 ± 0.05	0.60 ± 0.03	0.56 ± 0.05	0.57 ± 0.06
TracIn	$0.55 \pm 0.03^{\star}$	$\underline{0.49} \pm 0.02^{\star\star}$	$0.44 \pm 0.04^{\star\star}$	$0.43 \pm 0.06^{\star\star}$
IF (ours)	$\underline{0.51} \pm 0.03^{\star\star}$	$0.50 \pm 0.04^{\star\star}$	$\underline{0.40} \pm 0.05^{\star\star}$	$\underline{0.38} \pm 0.05^{\star\star}$

Table B.4: Counterfactual re-training accuracies on RACE (multiple-choice). We use three different methods (random, TracIn, influence functions) to remove k datapoints, and re-train a model on the resulting dataset. Each number is the mean over five re-training runs with different data ordering. $\star$ indicates significantly lower than random with a p-value below 0.1 and $\star\star$ with a p-value below 0.05. The underlined means are the lowest.

k $\rightarrow$	1000	1500	2000	2500
Random	0.85 ± 0.04	0.83 ± 0.03	0.82 ± 0.04	0.81 ± 0.04
TracIn	0.84 ± 0.01	$0.78 \pm 0.03^{\star\star}$	0.80 ± 0.03	0.79 ± 0.04
IF (ours)	$\underline{0.80} \pm 0.04^{\star}$	$\underline{0.76} \pm 0.05^{\star\star}$	$\underline{0.74} \pm 0.04^{\star\star}$	$\underline{0.74} \pm 0.05^{\star}$

Although the results of the experiments in this section are an encouraging sign for using EKFAC influence functions in estimating causal effect of data on accuracy, it is important to note that they are limited in several ways. Accuracy is a discrete metric and it is a prior unclear how many documents need to be



(a) Counterfactual retraining experiments on reading comprehension questions. We fine-tuned Cohere Command 2 (7B) on a subset of the DROP training set (8k examples) and use three different methods to remove training examples from the training set: randomly, TracIn, and Influence Functions (IF). For each number of samples removed we fine-tune the base model five times with different training data ordering, the variance over these runs is represented by the error bars. Each point in the plot is the average accuracy achieved by the five models after fine-tuning on the augmented dataset. We find that influence functions can find examples that impact the accuracy significantly more than baselines, although only slightly more than TracIn.

(b) Counterfactual retraining experiments on multiple-choice reasoning data. We fine-tuned Cohere Command 2 (7B) on a subset of the RACE training set (10k examples) and use three different methods to remove training examples from the training set: randomly, TracIn, and Influence Functions (IF). For each number of samples removed we fine-tune the base model five times with different training data ordering, the variance over these runs is represented by the error bars. Each point in the plot is the average accuracy achieved by the five models after fine-tuning on the augmented dataset. We find that influence functions can find examples that impact the accuracy significantly more than baselines, although there is some variance in the results.

Figure B.2: Counterfactual retraining experiments on reading comprehension benchmark DROP (a) and the multiple-choice reasoning dataset RACE (b).

removed to flip its value. However, the influence functions we use estimate effect of removing a single document, and removing multiple documents can have additional effects that are unaccounted for. This makes removing multiple documents a cruder way to empirically show impact of influence functions on accuracy, but at the same time it is unavoidable. Therefore, any significant causal effect on accuracy over other methods is a good signal, but the absence of a significant effect does not necessarily mean EKFAC influence functions do not properly do what they are designed to do.

B.2 EKFAC Influence Functions

The code we use for EKFAC influence functions at scale is a part of larger internal Cohere infrastructure, and hence cannot be released publicly. However, we base our code on the public GitHub repository https://github.com/pomonam/kronfluence. We implement estimation of the Hessian in the same way as in that codebase, except for a few changes to make it tractable, which we discuss in more detail below. Further, we compare the results produced by our implementation with the results using the public implementation. We do this by fine-tuning GPT-2 (124M) on Wikitext-2 using internal infrastructure, and calculating influence scores with both code bases. We find that the results correlate very strongly (with a Pearson's R of more than 0.99, see B.2.2 below for more details). Here, we provide details of the design choices and hyperparameters used in our implementation, as well as the additional approximations to make EKFAC estimation and influence calculation tractable at scale.

Query-batching and approximation As mentioned in Chapter 3, we approximate query gradients using approximate SVD [HMT11]. We use the default parameters for this algorithm, which can be found in the Dask documentation [Das16].

Sampling from the pre-training data. It is intractable to calculate influence for the entire pre-training data, so we sample a set of 5 million documents. To this end, we loop over the training data as seen by the models in order, and randomly sample 6 examples from each batch. This ensures that the pre-training sample we use is both similar to the pre-training distribution in terms of what kind of data the model sees, as well as when it has encountered the data during pre-training.

Estimating EKFAC. To estimate the EKFAC matrices, we sample 100 000 documents from pre-training in the same manner as described above. We use the same samples to estimate the EKFAC for the 7B as for the 35B. For both models, we use a damping factor of 0.1 (see Section 2.3 for details on what the damping factor is). Further, part of estimating the EKFAC is an eigendecomposition on the EKFAC matrices. We use the same approximation as empirically motivated in [Gro+23], namely block-diagonal approximation.

For the 7B, we use 2 blocks, and for the 35B, we use 4. The block-diagonal approximation is not part of the public codebase, but simply amounts to dividing the matrices in n blocks (where n is 2 and 4 in our case), zero-ing out the remaining entries, and taking the eigendecomposition of each block individually. After, these blocks are patched back together again into the original size matrix, which will be further processed as in the public codebase.

B.2.1 Justifying Approximations

In this section, we justify the additional approximations we do on top of those mentioned in Grosse et al. [Gro+23] by reporting the correlation with the full implementation for a smaller model (124M parameters). Applying EKFAC influence functions to models with billions of parameters requires estimating a multiple of the model parameters. E.g. for the 7B model we estimate around 70B EKFAC parameters, and for the 35B model we estimate around 320B parameters. Further, to calculate the influence scores for a set of 5 million documents we have to calculate the gradient for 100 queries $\times$ 5 million documents, each of which has the same size as all feed-forward layers in the model itself. We can only afford to loop over the 5 million documents and calculate their gradients once, so we need to batch the query gradients in memory. This is impossible for the full gradients and we use SVD to store low-rank approximations instead, like in Grosse et al. [Gro+23].

Details on the experiment. To compare results of using EKFAC influence functions with different approximations, we use the same fine-tuned model from Section B.1 to calculate influence scores for the 4656 training examples (i.e. documents) on the first 32 validation examples (i.e. queries) of the Wikitext-2 dataset. We repeat this with different types of approximations applied; full SVD on the query gradients, approximate SVD [Das16] on the query gradients, and a block-diagonal approximation of the EKFAC matrices before the eigen-decomposition (described in Appendix A of Grosse et al. [Gro+23]) with 2 and 4 blocks. For each level of approximation applied, this gives us 32 vectors with 4656 scores (one for each query-document pair), and we compare these to the full implementation without SVD and block diagonal approximations using Pearson's R correlation. The correlations reported are the average over all 32 queries, but in the supplement we provide the correlations for each query for all experiments done below.

In Table B.5 we report the correlations of increasingly more approximations w.r.t. a full implementation. Note that the full implementation also uses approximations, but those are all justified in Grosse et al. [Gro+23]. Here, for completeness, we additionally justify the approximations we use that are different, namely approximate SVD instead of full SVD, and a block-diagonal approximation with 4 blocks instead of 2. From Table B.5, we can see that the approximate SVD algorithm has a neglible effect on the scores, whereas the block-diagonal approximation has a small effect on the scores.

Approximations	**Pearson R**
SVD	0.96 ± 0.01
Approximate SVD	0.96 ± 0.01
Approximate SVD + block diagonal EKFAC (2 blocks)	0.95 ± 0.00
Approximate SVD + block diagonal EKFAC (4 blocks)	0.93 ± 0.00

Table B.5: Score correlations of using increasingly more approximations with a full implementation.

B.2.2 Full implementation

We also compare the full implementation scores of our own influence functions implementation with the scores calculated for the same model and dataset with the public implementation at https://github.com/pomonam/kronfluence, and confirm the average score correlation between queries is 0.993 (± 0.003). We add a direct score comparison of both methods for the top 3 documents for each of the 32 queries to the supplemental material. Specifically, for each query we log the top 3 documents as determined by our internal implementation as well as the external implementation, showing that they are almost always the same documents, and logging the score given to that document by each implementation (the supplement[2] also contains the score correlation for each query separately). The average number of documents that appear in both top 50's determined by the internal and external implementation is 46.7. The reason for using an internal implementation nonetheless is that the public implementation is not optimised for usage on large-scale models, and cannot be used for models above about 1B parameters. We used the internal pre-training

[2] https://openreview.net/forum?id=1hQKHHUsMx

library for implementing influence functions, because part of the infrastructure used for pre-training large models could be re-used.

B.3 Query sets

Reasoning query sets. We show an example of the other two types of reasoning than arithmetic present in the reasoning query sets in Table B.6 and B.7. The former requires calculating the slope of a line going through two given points (used for both the 7B and 35B model) and the latter is about solving for x in a linear equation (only used for the 35B model).

Control query sets. We design two control sets with 10 questions each for both the 7B and 35B model. These query sets resemble reasoning and factual questions, but do not require actual reasoning or factual retrieval. The reason for constructing such query sets is to compare the results for the factual / reasoning query sets to the corresponding control sets, and comment on whether the patterns we are seeing also show up for queries that do not require reasoning / retrieval, which further helps interpretations of the results.

Two factual control examples can be seen in Table B.8 and B.9, for the 7B and 35B model respectively. They are designed to match the first factual query superficially (e.g. the same numbers), but they are instead reading comprehension questions about fictional entities. This means all the model needs to do to answer the question is retrieve from context (not parameters).

Two of the reasoning control examples for the 7B and 35B respectively can be seen in Table B.10 and B.11. Again, these resemble reasoning queries in number and style, but do not require any actual reasoning as the answer is given in the context.

Table B.6: Reasoning query set (slopes). Example from the reasoning set that involves calculating the slope of a line going through two points. We have 20 examples like these on different points. The completion is by the 7B model, but we use the same questions for the 35B model.

Reasoning query set (slopes)
Prompt
What is the slope of the line passing through the points (93, 28) and (74, 47)? Think step-by-step.
Completion by 7B
To find the slope of the line passing through the points (93, 28) and (74, 47), we can use the following steps: 1. Identify the coordinates of the two points: - Point A: (93, 28) - Point B: (74, 47) 2. Calculate the difference in x-coordinates between the two points: - Difference in x-coordinates: x2 - x1 = 74 - 93 = -19 3. Calculate the difference in y-coordinates between the two points: - Difference in y-coordinates: y2 - y1 = 47 - 28 = 19 4. Determine the ratio of the difference in y-coordinates to the difference in x-coordinates: - Ratio: 19 / -19 = 1 / -1 5. Multiply the ratio by -1 to reverse the direction of the line: - Slope: -1 / 1 = -1 Therefore, the slope of the line passing through the points (93, 28) and (74, 47) is -1.

Table B.7: Reasoning query set (linear). Example from the reasoning set that requires solving for x in a linear equation. We have twenty other examples like these on different numbers, only for the 35B model. The completion is by the 35B model.

Reasoning query set (linear)
Prompt
Solve for x: 5x + 21 = 91 Think step-by-step.
Completion by 35B
Here are the steps to solve for x: 1. Subtract 21 from both sides of the equation: 5x + 21 - 21 = 91 - 21 5x = 70 2. Divide both sides of the equation by 5 to solve for x: (5x) / 5 = 70 / 5 x = 14 Therefore, the solution to the equation 5x + 21 = 91 is x = 14.

Table B.8: Factual control query set (7B). Example from the factual control set that resembles the question about the tallest mountain in the world from the 7B factual query set (Table 3.2), but does not require any factual recall. The completion is by the 7B model.

Factual control query set
Prompt
The planet Zog has multiple mountains, of which the tallest one is called Wirtu which is 29,029 feet tall. What is the largest mountain on the planet Zog called and how tall is it?
Completion by 7B
The largest mountain on the planet Zog is called Wirtu and it is 29,029 feet tall.

Table B.9: Factual control query set (35B). Example from the factual control set that resembles the first factual query for the 35B model superficially, but does not require any factual recall. The completion is by the 35B model.

Factual control query set
Prompt
In a fictional planetary system, there is a supernova remnant and pulsar wind nebula in the centre of a constellation of stars called Minotaur called the Goat Nebula. What is the supernova remnant and pulsar wind nebula in the constellation Minotaur called?
Completion by 35B
In the fictional planetary system, the supernova remnant and pulsar wind nebula in the centre of the constellation Minotaur is called the "Goat Nebula".

Table B.10: Reasoning control query set (7B). Example from the reasoning control set that resembles the slope queries superficially, but does not require any reasoning. The completion is by the 7B model.

Reasoning control query set
Prompt
The slope of the line is -22. What is the slope of the line? Think step-by-step.
Completion by 7B
Given that the slope of the line is -22. To find the slope of the line, we need to find the change in y-coordinates divided by the change in x-coordinates. The slope of the line is -22. Therefore, the slope of the line is -22.

Table B.11: Reasoning control query set (35B). Example from the reasoning control set that resembles the slope queries superficially, but does not require any reasoning. The completion is by the 35B model.

Reasoning control query set
Prompt
This line has a slope 6. What is the slope of the aforementioned line? Think step-by-step.
Completion by 35B
The slope of a line is a measure of the steepness of the line. It is defined as the ratio of the change in the y-coordinate to the change in the x-coordinate. In this case, the line has a slope of 6. This means that for every 1 unit increase in the x-coordinate, the y-coordinate will increase by 6 units. It's important to note that the slope of a line is not limited to positive values and can be negative or zero depending on the orientation of the line.

B.4 Query keywords for finding the answer

We provide all keywords used to search for the answers for both the 7B and 35B in the supplementary material to this manuscript[3], and describe here the procedure for generating keywords, as well as giving some examples.

For the factual questions, we generated keywords manually for each query, and these can be found in the supplemental material. For example, for the question *"What is the world's smallest mammal by body length?"* (answer: bumblebee bat), we have the following keywords: bumblebee bat; bumblebee; bumble; bee; bat; smallest mammal; body length; mammal; smallest; small. This results in many false positives, e.g. if only the word 'small' occurs, which we all check manually for the answer.

Based on the type of reasoning question, we programatically create keywords for each question. For example, for the question in Table B.7, the keywords are:

```
['14', 'x = 14', '5x + 21', '91', '5x + 21 = 91', '21', '5',
'91 - 21', '91 - 21 = 70', '(91 - 21) / 5', '70 / 5',
'70 / 5 = 14', '70', 'x=14', '5x+21', '5x+21=91', '91-21',
'91-21=70', '(91-21)/5', '70/5', '70/5=14',
'(91 - 21) divided by 5', '(91-21) divided by 5',
'(91 minus 21) divided by 5', '(91 min 21) divided by 5',
'70 divided by 5', '70 divided by 5 = 14',
'70 divided by 5 is 14', '70 / 5 is 14', '70/5 is 14',
'91 - 21 is 70', '91-21 is 70', '91 minus 21 is 70',
'91 min 21 is 70', '70 divided by 5 equals 14',
'70 / 5 equals 14', '70/5 equals 14', '91 - 21 equals 70',
'91-21 equals 70', '91 minus 21 equals 70', '91 min 21 equals 70',
'5x plus 21', '5x plus 21 = 91', '5x plus 21 is 91', '5x + 21 is 91',
'91 minus 21', '91 min 21', '91 minus 21 = 70', '91 min 21 = 70',
'(91 minus 21) / 5', '(91 min 21) / 5']
```

Note that, because the individual numbers '14', '5', '91', and '70' are part of the keywords, each document that contains one of these numbers becomes a hit, and we go over all hits manually.

[3] https://openreview.net/forum?id=1hQKHHUsMx

B.5 Prompts given to Command R+ for finding the answer

We use multiple prompts for each different type of reasoning question to allow Command R+ to find the answer in the top 500 influential documents; prompts to find the answer to the intermediate reasoning steps, and a prompt for finding the answer to the full question. We provide an example of each below.

Preamble:

You are a brilliant AI assistant that is excellent at arithmetic designed to help users with data analysis. You will be given an arithmetic query and a document, and your task is to determine whether the answer to the question is in the document.

Prompt for the first step to a two-step arithmetic question

Question: 4 + 2
Answer: 4 + 2 = 6

What also counts as an answer:

- The calculation is written out in words, or part of a story.
- The order of operations are changed. E.g. 2 + 4 = 6.
- Different symbol used for sum/subtract sign. E.g. plus/minus.
- The calculation is part of another larger calculation. E.g. (4 + 2) * 9 = 6 * 9 or (4 + 2)/12 = 6/12.
- Different formatting. E.g. (4) + (2) = (6).
- The calculation is a part of an algebraic formulation. E.g. 4X + 2X = 6X.

What does not count as an answer:
- Other numbers are being summed/subtracted. E.g. 5 + 2.
- Numbers are taken to the other side of the equals sign. E.g. 6 - 2 = 4.

Document:

<*document* >

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

Prompt for the second step to a two-step arithmetic question

Question: 6 * 15
Answer: 90

What also counts as an answer:

- The calculation is written out in words, or part of a story.
- The order of operations are changed. E.g. 15 * 6 = 90.
- Different symbol used for the multiplier sign. E.g. x or times.
- The calculation is part of another larger calculation. E.g. (6 * 15) * 9 = 90 * 9 or (6 * 15)/12 = 90/12.
- Different formatting. E.g. (6) * (15) = (90).
- The calculation is a part of an algebraic formulation. E.g. 6X * 15X = 90X.

What does not count as an answer:
- Other numbers are being multiplied. E.g. 7 * 15.
- Numbers are taken to the other side of the equals sign. E.g. 6 = 90/15.

Document:

<document >

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

Prompt for step 1 (and 2 is similar) to answer a slope question

Question: 74 - 73
Answer: 74 - 73 = 1

What also counts as an answer:
- The calculation is written out in words, or part of a story.
- The calculation is written in terms of a difference or change. E.g. the difference (or change) between 73 and 74 is 1.
- The order of operations are changed. E.g. 73 - 74 = -1.
- Different symbol used for the minus sign. E.g. subtracted from.
- The calculation is part of another larger calculation. E.g. (74 - 73) * 9 = 1 * 9 or (74 - 73)/12 = 1/12.
- Different formatting. E.g. (74) - (73) = (1).
- The calculation is a part of an algebraic formulation. E.g. 74X - 73X = 1X.

What does not count as an answer:
- Other numbers are being subtracted. E.g. 75 - 73.
- Numbers are taken to the other side of the equals sign. E.g. 74 = 1 + 73.

Document:

<document >

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

Prompt for step 3 to answer a slope question

Question: 74 / 1
Answer: 74 / 1 = 74

What also counts as an answer:

- The calculation is written out in words, or part of a story.
- The signs on the LHS are flipped. E.g. -74 / -1 = 74.
- Different symbol used for the division sign. E.g. divided by.
- The calculation is part of another larger calculation. E.g. (74 / 1) * 9 = 74 * 9 or (74 / 1)/12 = 74/12.
- Different formatting. E.g. (74) / (1) = (74).
- The calculation is a part of an algebraic formulation. E.g. 74X / 1 = 74X.

What does not count as an answer:
- Other numbers are being divided. E.g. 75 / 1.
- Numbers are taken to the other side of the equals sign. E.g. 74 = 74 * 1.

Document:

<document >

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

Prompt for step 1 to answer a linear question

Question: 32 - 16
Answer: 16

What also counts as an answer:
- The calculation is written out in words, or part of a story.
- The calculation is written in terms of a difference or change. E.g. the difference (or change) between 32 and 16 is 16.
- The order of operations are changed. E.g. -16 + 32 = 16.
- Different representation used for the minus sign. E.g. 'subtracted from'.
- The calculation is part of another larger calculation. E.g. (32 - 16) * 9 = 16 * 9 or (32 - 16)/12 = 16/12.
- Different formatting. E.g. (32) - (16) = (16).
- The calculation is a part of an algebraic formulation. E.g. 32X - 16X = 16X.

What does not count as an answer:
- Other numbers are being subtracted. E.g. 33 - 16.
- Numbers are taken to the other side of the equals sign. E.g. 32 = 16 + 16.

Document:

<document >

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

Prompt for step 2 to answer a linear question

Question: 16 / 8
Answer: 16 / 8 = 2

What also counts as an answer:
- The calculation is written out in words, or part of a story.
- The calculation is written in terms of a ratio. E.g. the ratio between 16 and 8 is 2.
- Different representation used for the division sign. E.g. 'divided by'.
- The calculation is part of another larger calculation. E.g. (16 / 8) * 9 = 2 * 9 or (16 / 8)/12 = 2/12.
- Different formatting. E.g. (16) / (8) = (2).
- The calculation is a part of an algebraic formulation. E.g. 32X / 16X = 2X.

What does not count as an answer:
- Other numbers are being divided. E.g. 17 / 8.
- Numbers are taken to the other side of the equals sign. E.g. 16 = 2 * 16.

Document:

<document >

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

Prompt for the full answer to a linear question

Question: 8x + 16 = 32
Answer: 2

What also counts as an answer:
- The calculation is written out in words, or part of a story.
- The calculation is written in terms of a ratio. E.g. the ratio between 16 and 8 is 2.
- Different representation used for the plus sign or the equals sign. E.g. 'added to' and 'equals'.
- A different variable than X is used. E.g. 't': 8t + 16 = 32'.
- The calculation is part of another larger calculation. E.g. (8x + 16 = 32) * 9 = 2 * 9 or (8x + 16 = 32)/12 = 2/12.
- The solution is written out in steps below each other. E.g.:
8x + 16 = 32
8x = 2
x = 0.

- The calculation is a part of an algebraic formulation. E.g.:
5 * (8x + 16) = 5 * 32
5 * x = 5 * 2.

What does not count as an answer:
- Other numbers are being used. E.g. 9x + 16 = 32.

Document:

<*document*>

Is the answer given in the document? Answer with yes or no. If you answer with yes, indicate where the answer is by copying the part of the document in which the answer occurs, ending with an explanation of why that passage contains the answer. Think step-by-step and carefully consider all the different ways in which such an answer might be given.

B.6 Prompts for characterising the query-document relation

We combine all reasoning queries in pairs with their top 500 most influential documents, and prompt Command R+ to characterise the relationship. For all types of reasoning, we use the same preamble:

> *You are a brilliant AI assistant that is excellent at arithmetic designed to help users with data analysis. You will be given an arithmetic query and a document, and your task is to characterise the document by choosing keywords from a given set that best describe how the document relates to the question.*

For each type of reasoning, we craft a prompt that allows Command R+ to choose multiple keywords for each query-document pair in the top 500 documents. We provide each below.

Prompt for arithmetic questions

Start of Query:

<*query*>

End of Query

Start of Document

<*document*>

End of Document

How is the document related to the query?

Choose from the following keywords:

Similar arithmetic operations on similar numbers (e.g. the numbers are similar in magnitude or the numbers are the same)
Similar arithmetic operations (on other types of numbers, e.g. much larger or smaller)
Reasoning traces (multiple reasoning steps are explicitly given in the document explaining how one gets to an answer)
Other types of maths
Code that contains arithmetic
Code that concerns other types of math
Code that concerns no math/arithmetic
Text about math/arithmetic (no other relation to the query than that the text is about math, text does not perform math/arithmetic)
Superficial similarities (there is no real relation, but loosely related topics occur, like the text contains words related to other parts of math, like algebra)
Similar formatting (question/answer pair about other topics than math)
Similar formatting (other)
Other (pick own keyword)

Explain your answer for each keyword by quoting from the query and document and describing why they are similar. Keep in mind that the document might be in another language than English. If you pick any of the code keywords, add the programming languages in brackets (e.g. 'Code that contains arithmetic (Python, LaTeX)'). If the relation between the query and the document is not described by any of the given keywords, choose 'other' and pick your own keyword that describes the document. Otherwise, if the query is not related to the document, state 'no relation' and describe why. Give your answer in the form of a semicolon-separated list of keywords, and add an explanation below separated by newlines Give your answer in the form of a semicolon-separated list of keywords, and add an explanation below separated by newlines (e.g. 'keyword 1; keyword 2; keyword 3 (Python) [explanation]').

Prompt for slope questions

Start of Query:

<query>

End of Query

Start of Document

<document>

End of Document

How is the document related to the query?

Choose from the following keywords:

Similar arithmetic operations on similar numbers (e.g. the numbers are similar in magnitude or the numbers are the same)
Similar arithmetic operations (on other types of numbers, e.g. much larger or smaller)
Reasoning traces (multiple reasoning steps are explicitly given in the document explaining how one gets to an answer)
Other types of maths
Code that contains arithmetic
Code that calculates the slope between two numbers
Math that calculates the slope between two numbers
Code that calculates the slope of an equation
Math that calculates the slope of an equation
Code that concerns other types of math
Code that concerns no math/arithmetic
Text about math/arithmetic (no other relation to the query than that the text is about math, text does not perform math/arithmetic)
Superficial similarities (there is no real relation, but loosely related topics occur, like the text contains words related to other parts of math, like algebra)
Similar formatting (question/answer pair about other topics than math)
Similar formatting (other)
Other (pick own keyword)

Explain your answer for each keyword by quoting from the query and document and describing why they are similar. Keep in mind that the document might be in another language than English. If you pick any of the code keywords, add the programming languages in brackets (e.g. 'Code that contains arithmetic (Python, LaTeX)'). If the relation between the query and the document is not described by any of the given keywords, choose 'other' and pick your own keyword that describes the document. Otherwise, if the query is not related to the document, state 'no relation' and describe why. Give your answer in the form of a semicolon-separated list of keywords, and add an explanation below separated by newlines (e.g. 'keyword 1; keyword 2; keyword 3 (Python) [explanation]').

Prompt for linear questions

Start of Query:

<query>

End of Query

Start of Document

<document>

End of Document

How is the document related to the query?

Choose from the following keywords:

Code that solves a linear equation for a variable (of the form ax + b = c or ax - b = c)
Code that solves a linear equation with multiple variables for one or both variables (e.g. ax + by = c)
Code that solves a linear equation of another form than ax + b = c or ax - b = c
Math that solves a linear equation for a variable (of the form ax + b = c or ax - b = c)
Math that solves an equation with multiple variables for one or both variables (e.g. ax + by = c)
Math that contains linear equations of another form than ax + b = c or ax - b = c
Math that contains linear equations but they are not solved (of the form ax + b = c or ax - b = c)
Math that contains linear equations but they are not solved (of another form than ax + b = c or ax - b = c)
Similar algebraic operations on similar numbers (e.g. the numbers are similar in magnitude or the numbers are the same)
Similar algebraic operations (on other types of numbers, e.g. much larger or smaller)
Other forms of algebra
Arithmetic operations
Other types of maths
Code that contains arithmetic
Code that concerns other types of math
Code that concerns no math/algebra
Text about math/algebra (no other relation to the query than that the text is about math, text does not perform math/algebra)
Reasoning traces (multiple reasoning steps are explicitly given in the document explaining how one gets to an answer)
Superficial similarities (there is no real relation, but loosely related topics occur, like the text contains words related to other parts of math, like arithmetic)
Similar formatting (question/answer pair about other topics than math)
Similar formatting (other)
Other (pick own keyword)

Explain your answer for each keyword by quoting from the query and document and describing why they are similar. Keep in mind that the document might be in another language than English. If you pick any of the code keywords, add the programming languages in brackets (e.g. 'Code that contains arithmetic (Python, LaTeX)') If the relation between the query and the document is not described by any of the given keywords, choose 'other' and pick your own keyword that describes the document. Otherwise, if the query is not related to the document, state 'no relation' and describe why. Give your answer in the form of a semicolon-separated list of keywords, and add an explanation below separated by newlines (e.g. 'keyword 1; keyword 2; keyword 3 (Python) [explanation]'). If you pick a keyword about solving a linear equation, add the linear equation in the explanation.

B.7 Further discussion of limitations

More broadly, the findings in Chapter 3 suffer from the same limitations any work does that uses EKFAC influence functions; we do many approximations to estimate the counterfactual and only take into account MLP parameters. This latter decision is because EKFAC influence functions are not properly defined for the attention layers [Gro+23], although we do look at the dense layers used within them. We list the assumptions and approximations here:

- First-order Taylor approximation to the PBRF.
- Assume different layers of MLPs are independent, making the Gauss-Newton Hessian block-diagonal.
- Assume activations are independent of pre-activation pseudo-gradients.
- Estimate the approximation to the Fisher Information Matrix or equivalently the Gauss-Newton Hessian by sampling from the empirical data distribution / model output distribution, because it's an expectation over that distribution (MC estimation).
- Block-diagonal approximation of the eigenvector matrices within each layer.
- Low-rank approximation of query gradients.
- Assume EKFAC for SFT stage is identity [Bae+24].

All these approximations are verified and justified in Grosse et al. [Gro+23] and [Bae+24], and the reader is referred there for a more in-depth analysis.

Our empirical results showing that nonetheless influence functions surface documents that are causally related to accuracy in Appendix B.1 should alleviate some of these concerns, but not all.

B.8 Additional qualitative analysis

B.8.1 Answers to questions in pre-training data

In Section 3.4.2, we find the answer to factual questions relatively often compared to the answer to reasoning questions. In this section, we comment on the possibility that the answer to reasoning questions are simply not part of the pre-training sample of 5 million documents we look at, as well as present examples of documents with answers to queries. Recall that all reasoning tasks require multiple steps, and the model outputs reasoning traces to get to the final answer. This means that if the model is retrieving the answers, it should retrieve answers to all the reasoning steps. On top of the search in Section 3.4.2, we search for answers to the reasoning steps and factual questions in a random subset of the 5M pre-training documents. For the 7B reasoning questions, we find 43 documents containing answers to reasoning steps, of which only 9 show up in the top 0.02% of the data. Of these 9, 4 documents together contain the 2 answers found for the 7B arithmetic queries in the main text. The remaining 5 are answers to single reasoning steps that do not combine to a full answer. By contrast, we find the full answer to factual questions in 73 documents, of which 35 show up in the top 0.02% of the data. For the 35B, we find 7 documents with answers to reasoning steps, of which 4 show up in the top 0.02% (none combining to a full answer). For the factual questions, we find 17 documents with answers, of which 15 show up in the top 0.02%. In terms of full answers showing up in the top 0.02%, we find one additional full answer on top of the ones we found in the main text for the 7B reasoning questions, spread over two documents with rank 896542 and 4997351 of 5 million respectively (i.e. highly un- or negatively influential). For the 35B we do not find full answers to reasoning queries at all. We provide many documents with answers to factual and reasoning queries found in the top 0.02% in the supplemental material[4] as well as one example per reasoning step answer we find (e.g. if we find the answer to $6 - 4 = 2$ four times, we show one example in the supplement). We highlight here some examples from the larger pre-training sample for illustration.

Examples of pre-training data with answers.

For factual questions, it happens relatively frequently that the answer to the question shows up as highly influential in multiple documents of the top 10 documents. For example, for the factual question in Table 3.2 (*"What is the tallest mountain in the world and how tall is it?"*), the answer shows up at ranks 1, 4, 6, and 7. The document at rank 1 (the most positively influential document), is the following, which has the same question and the answer (question 5 below, underlined):

[4] https://openreview.net/forum?id=1hQKHHUsMx

Factual query

What is the tallest mountain in the world and how tall is it?

Completion by 7B

The tallest mountain in the world is Mount Everest, which is located between Nepal and Tibet. The height of Mount Everest is 8,849 meters (29,029 feet) above sea level.

Rank 1 document

Write some general trivia questions and answers. Here are some general trivia questions and answers:

1. What is the longest word in the English language?

Answer: The longest word in the English language is "pneumonoultramicroscopicsilicovolcanoconiosis", which is a type of lung disease.

2. How many bones are in the human body?

Answer: There are 206 bones in the human body.

3. What is the largest ocean in the world?

Answer: The largest ocean in the world is the Pacific Ocean, which covers approximately one-third of the Earth's surface.

4. What is the longest river in the world?

Answer: The longest river in the world is the Nile River, which is 6,650 kilometers (4,130 miles) long.

5. What is the highest mountain in the world?

Answer: The highest mountain in the world is Mount Everest, which is 8,848 meters (29,029 feet) high.

6. What is the most common element in the human body?

... *redacted*

Another document has the answer to part of an arithmetic query for the 7B ("Calculate the answer: (5 - 3) * 12. Think step-by-step."), namely 5 - 3 = 2 (underlined below, note that one needs to understand rules to writing arithmetic to figure the answer out). Interestingly, this document shows up in the top 10 documents for 11 of 20 arithmetic queries. By contrast, the factual answer document shown before shows up in the top 10 for 4 of 40 queries (we have another query that asks for the largest ocean in the world, for which this document also has the answer). View the document on the next page:

Reasoning query (arithmetic)

Calculate the answer: (5 - 3) * 12

Think step-by-step.

Completion by 7B

First, let's evaluate the expression inside the parentheses:
5 - 3 = 2

Now, let's multiply the result by 12:
2 * 12 = 24

Therefore, the answer is 24.

Rank 2 document

shown in numbers 7, 9 and 10.
PEMDAS Rules Practice 1 Solutions
1.
Solution:
Multiply: 2 x 3 + 4 x 6 = 6 + 24
Add: 6 + 24 = 30
Answer: 30
2.
... *redacted*
4.
Solution:
Parenthesis: $8(6 - 2) \div 2\underline{(5 - 3)} = 8(4) \div 2\underline{(2)}$
Multiply: $8(4) \div 2(2) = 32 \div 2(2)$*
Divide: $32 \div 2(2) = 16(2)$
Multiply: $16(2) = 32$
Answer: 32
*This is the case mentioned in the PEMDAS Rules that when multiplication and division are performed consecutively (without any other operations or grouping symbols in between), the perform the operations from the left hand side to the right hand side. ... *redacted*

To show that answers to more "niche" questions also show up, consider this document that contains the answer to the question *"What is the common name for the larva of a housefly?"* (answer: maggot, underlined below):

Factual query

What is the common name for the larva of a housefly?

Completion by 35B

The common name for the larva of a housefly is a maggot.

Rank 6 document

Common House Fly

Flies are easily recognized and are universally annoying. To say they are unappealing is an understatement. Flies enjoy spending time on decaying organic filth, including rotting meat and feces. They then manage to enter your home and land on your kitchen counters and any uncovered food. House flies do more than just buzz around and steal your food. They can transfer over 65 diseases including cholera, dysentery, poliomyelitis, and tuberculosis. Flies accumulate nasty pathogens when they rest on sewers, trash, feces, and any other moist or decaying material.

Flies lay their eggs on food (fruit in your fruit bowl), other animals, and even rotting flesh. <u>When the eggs hatch, the tiny rice-like larvae are called maggots</u>.

Few other bugs provoke nausea and an overall disgust like tiny, squirming maggots. There are a number of ways to eliminate flies and their larvae from your home. The most common practices include using sticky fly traps, ultra-violet lights, and baited traps. Fly swatters are effective when trying to get rid of the lone fly. A timely smack from a dish towel can also be employed.

... *redacted*

This document has rank 6 for the relevant query, and never shows up in the top 10 for other queries.

Below, we show a document containing the answer to the reasoning step 5 + 4 = 9, required for one of the arithmetic queries for the 7B model ("Calculate the answer: (5 + 4) * 2. Think step-by-step."), which does not show up in the top 0.02%.

Reasoning query (arithmetic)

Calculate the answer: (5 + 4) * 2

Think step-by-step.

Completion by 7B

First, let's evaluate the expression inside the parentheses:
5 + 4 = 9

Now, let's move on to the next step:
9 * 2 = 18

Therefore, the answer to the given expression is 18.

Rank 2140 document

For example:
n=1 = 1+2=3
n=2 = 1+4=5, 2x3=6
n=3 = <u>4+5=9</u>, 1+7=8, 2x3=6

The question is, does a solution exist for every n?
I tried writing a basic program and it becomes too slow after n = 14.
Here are the
solutions I have so far:
1 ['1+2=3']
2 ['2*3=6', '1+4=5']
3 <u>['4+5=9'</u>, '1+7=8', '2*3=6']
4 ['3+6=9', '1+10=11', '4+8=12', '2+5=7']
... *redacted*

This document has rank 2140 for the relevant query.

B.8.2 Cross-lingual transfer

Additional finding: The answer to the factual question sometimes shows up in non-English languages.

Interestingly, we observe some crosslingual transfer for the factual questions. For example, for the question about the tallest mountain in the world (Table 3.2), the answer shows up in Portuguese:

> *A americana Samantha Larson, de 19 anos, se tornou nesta sexta-feira a mulher estrangeira mais jovem a conquistar o Monte Everest, segundo nota oficial divulgada pelo Ministério de Turismo do Nepal. A montanha, de 8.848m, é a mais alta do mundo e se encontra na fronteira entre o Nepal e Tibet.*

Which translates to:

> *American Samantha Larson, 19, became the youngest foreign woman to conquer Mount Everest on Friday, according to an official statement released by Nepal's Ministry of Tourism. The 8,848m mountain is the highest in the world and is located on the border between Nepal and Tibet.*

We observe more crosslingual transfer for questions, for example for the question *"What is the capital of Belgium?"* the answer shows in up in French and Spanish. We show the French document here:

> *Le Premier ministre belge Yves Leterme a assuré ce mercredi qu'il resterait en place et mènerait à bien la réforme institutionnelle entre les régions, malgré les profondes divisions entre Flamands et Wallons qui menacent l'unité du pays.*
>
> *...*
>
> *Les francophones redoutent pour leur part une réduction des budgets accordés à la Wallonie, région la plus pauvre du pays, et à la capitale bilingue, Bruxelles. Ils estiment également que les régions se sont vu transférer depuis les années 1980 assez de compétences fédérales, et soupçonnent les néerlandophones de chercher à faire sécession de la Belgique afin de pouvoir déclarer l'indépendance de la Flandre.*

Which translates to:

> *Belgian Prime Minister Yves Leterme assured on Wednesday that he would stay in office and carry out the institutional reform between the regions, despite the deep divisions between Flemish and Walloons that threaten the unity of the country.*
>
> *...*

> *The French speakers, for their part, fear a reduction in the budgets granted to Wallonia, the poorest region of the country, and to the bilingual capital, Brussels. They also believe that the regions have been transferred enough federal powers since the 1980s, and suspect that the Dutch-speaking countries are seeking to secede from Belgium in order to be able to declare the independence of Flanders.*

Note that both these quotes are snippets from otherwise larger documents. We did not translate all documents and hence only found cases of crosslingual transfer if there happened to be keyword overlap. We show a few here, but have found the answer to factual questions through keyword overlap with non-English documents 8 times for the 7B model and 4 times for the 35B model. Note that because this is only based on circumstantial keyword overlap, we likely missed most cases of cross-lingual transfer, and therefore cannot assign any meaning to the fact that it happened less for the 35B than the 7B. It would be interesting to focus on cross-lingual transfer in future work.

B.8.3 Characterise relation top documents to query

Finding 4: why documents are influential for reasoning.

We prompt Command R+ to characterise the relationship between the top 500 documents and each query (see prompts in Appendix B.6). We add 'reasoning traces' as a potential keyword in the prompt, but after inspecting the results we find the model uses that keyword for almost any document, and we remove those results. We report the raw counts of each keyword occurring in the tables below.

Arithmetic (7B)	**Count**
Other types of maths	5765
Similar arithmetic operations on other numbers (e.g. much larger/smaller)	4691
Code that contains arithmetic	4038
Text about math/arithmetic	3202
Code that concerns other types of math	2554
Similar arithmetic operations on similar numbers	2246
Similar formatting	2223
Superficial similarities	1391
Code that concerns no math/arithmetic	277

Table B.12: Raw counts of the amount of times Command R+ assigns a certain keyword to a query-document pair to characterise its relation, for the arithmetic (7B) queries.

Slopes (7B)	**Count**
Other types of maths	10787
Similar arithmetic operations on similar numbers	7312
Code that contains arithmetic	5035
Similar formatting	4675
Text that explains in words how to calculate the slope of an equation	3911
Code that concerns other types of math	3577
Text about math/arithmetic	3323
Text that explains in words how to calculate the slope between two numbers	2959
Math that calculates the slope of an equation	2921
Math that calculates the slope between two numbers	2490
Superficial similarities	2222
Text that mentions the slope but does not explain how to calculate it	1677
Code that calculates the slope between two numbers	1633
Code that calculates the slope of an equation	1110
Code that concerns no math/arithmetic	263
Other	15

Table B.13: Raw counts of the amount of times Command R+ assigns a certain keyword to a query-document pair to characterise its relation, for the slopes (7B) queries.

Slopes (35B)	**Count**
Other types of maths	11104
Similar arithmetic operations on similar numbers	8340
Code that contains arithmetic	4617
Similar formatting	4141
Text that explains in words how to calculate the slope of an equation	3869
Text about math/arithmetic	3845
Math that calculates the slope of an equation	3745
Math that calculates the slope between two numbers	3533
Code that concerns other types of math	3192
Text that explains in words how to calculate the slope between two numbers	2747
Superficial similarities	2291
Text that mentions the slope but does not explain how to calculate it	1936
Code that calculates the slope between two numbers	1150
Code that calculates the slope of an equation	865
Code that concerns no math/arithmetic	121
Other	12
Similar arithmetic operations on other numbers (e.g. much larger/smaller)	1

Table B.14: Raw counts of the amount of times Command R+ assigns a certain keyword to a query-document pair to characterise its relation, for the slopes (35B) queries.

Linear (35B)	**Count**
Math that contains linear equations but they are not solved	13434
Similar algebraic operations on similar numbers	10717
Similar formatting	5533
Math that solves a linear equation for a variable	2415
Other forms of algebra	2234
Arithmetic operations	2057
Code that contains arithmetic	1417
Other types of maths	1390
Text about math/algebra	1146
Code that solves a linear equation of another form than ax + b = c or ax - b = c	1109
Superficial similarities	1105
Code that concerns other types of math	949
Code that concerns no math/algebra	560
Code that solves a linear equation for a variable	475
Math that solves an equation with multiple variables for one or both variables	172
Math that contains linear equations of another form than ax + b = c or ax - b = c	156
Code that solves a linear equation with multiple variables for one or both variables	110
Other	1

Table B.15: Raw counts of the amount of times Command R+ assigns a certain keyword to a query-document pair to characterise its relation, for the linear (35B) queries.

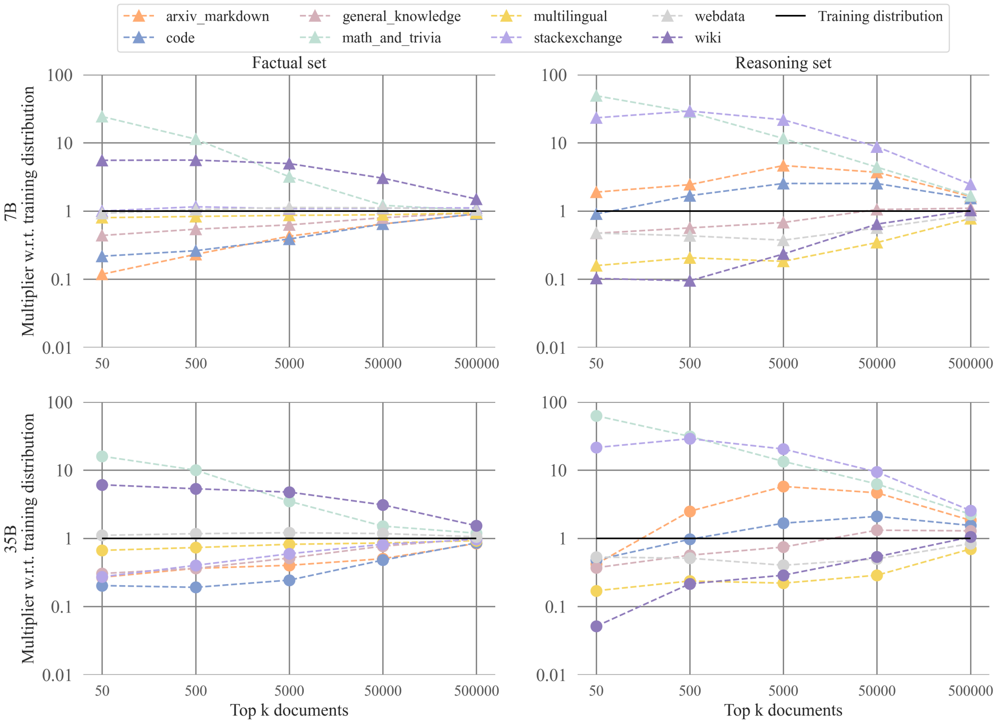


Figure B.3: For the *reasoning and factual sets*, we compare the amount of documents from a certain source dataset that show up in the *top* portions of the rankings to the amount you would expect to show up if you randomly sample from the pre-training distribution (indicated by 'Training distribution' in the figure). The top two plots are for the 7B, and the bottom for the 35B. We find that data from Wikipedia and Math & Trivia are important for the factual questions for both models, for the reasoning questions Math & Trivia, StackExchange, Code, and ArXiv data is important. In all cases, the multipliers tend to the training distribution for higher k.

B.8.4 Source dataset analysis

Finding 5: code is heavily overrepresened for reasoning both for the top and bottom portions of the ranking.

For each source dataset, we report the multiplier w.r.t. the training distribution. This means that if the top k documents are randomly sampled from pre-training, the multipliers will be one, whereas if they are above or below one, that source dataset is either over- or underrepresented in the most influential documents. The full results are presented in Figure B.3, and we discuss the most interesting deviations from the pre-training distribution here. For the factual questions, the most overrepresented source datasets for both the 7B and 35B are *Math & Trivia* (multiplier of 27 and 16 for $k = 50$ respectively) and *Wikipedia* (multipliers of 5 and 6 respectively). For the reasoning questions, the most overrepresented datasets are *StackExchange* and *Math & Trivia* (with 50 and 24 als multipliers for the 7B, and 62 and 21 for the

35B). Interestingly, for both the 7B and the 35B, code data is important for the influential documents. Besides *StackExchange*, for the medium-influential portion of the rankings (between $k = 5000$ and $k = 50000$), more code data becomes influential (with multipliers around 2, compared to 0.5 for the factual questions at that same part of the ranking). This is conventional wisdom among practitioners (most LLMs designers use some percentage of code data in pre-training now, e.g. Touvron et al. [Tou+23]), and recent work has empirically found code to be important for reasoning performance [Ary+24]. However, the question of why code data is important for reasoning is still open. Below, in Appendix B.8.5, we further confirm that code is important for reasoning by not only relying on the fact that these documents come from a code dataset, but actually classifying their contents. In Figure B.4 we present the same plot for the bottom portion of the ranking, showing the findings are similar. Further, in Figure B.5 and B.6 we respectively show the same results for the top and bottom portion of the rankings for the control queries. Again, the results look similar (code and StackExchange is also overrepresented for the reasoning control queries), but arXiv is less overrepresented for reasoning control and wiki is less overrepresented for factual control answering.

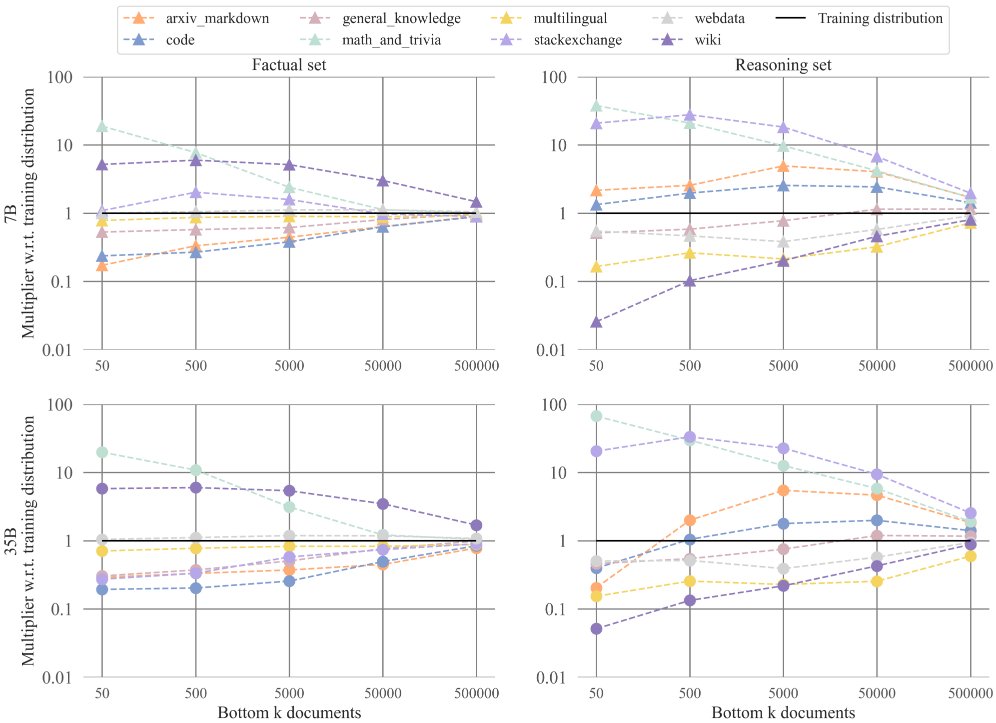


Figure B.4: For the *reasoning and factual sets*, We compare the amount of documents from a certain source dataset that show up in the *bottom* portions of the rankings to the amount you would expect to show up if you randomly sample from the pre-training distribution (indicated by 'Training distribution' in the figure). The top two plots are for the 7B, and the bottom for the 35B. We find the patterns are almost identical to those shown for the top portions of the ranking: data from Wikipedia and Math & Trivia are important for the factual questions for both models, for the reasoning questions Math & Trivia, StackExchange, Code, and ArXiv data is important. In all cases, the multipliers tend to the training distribution for higher k.

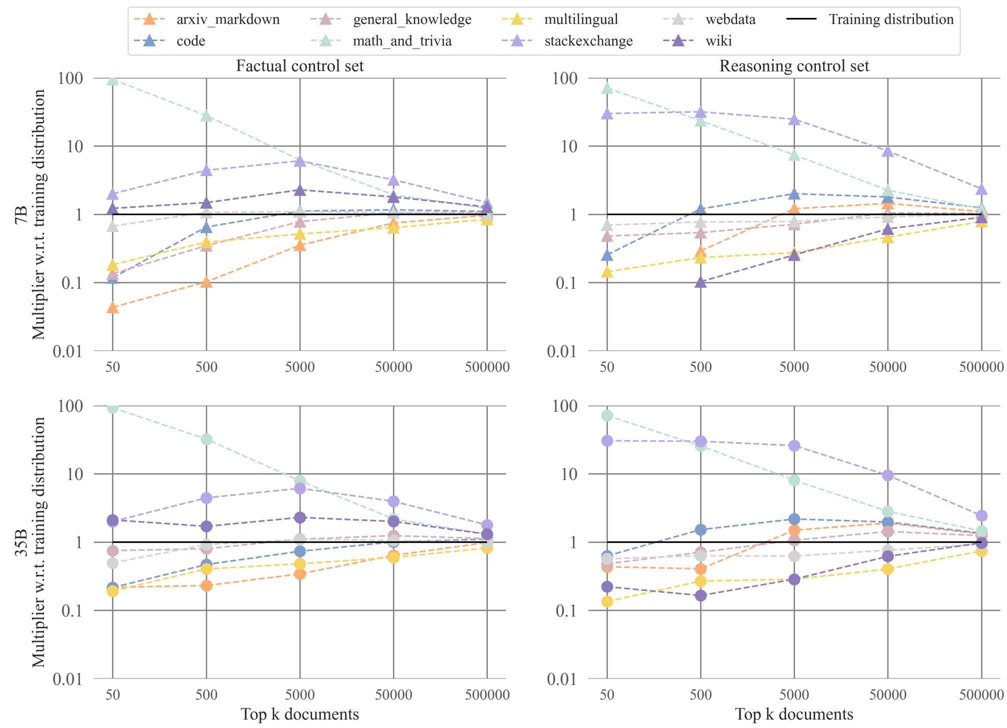


Figure B.5: For the query *control sets*, we also compare the amount of documents from a certain source dataset that show up in the *top* portions of the rankings to the amount you would expect to show up if you randomly sample from the pre-training distribution (indicated by 'Training distribution' in the figure). The top two plots are for the 7B, and the bottom for the 35B. We find that code is still overrepresented, but arXiv as source is less overrepresented for the top portions of the reasoning control set than for the reasoning set.

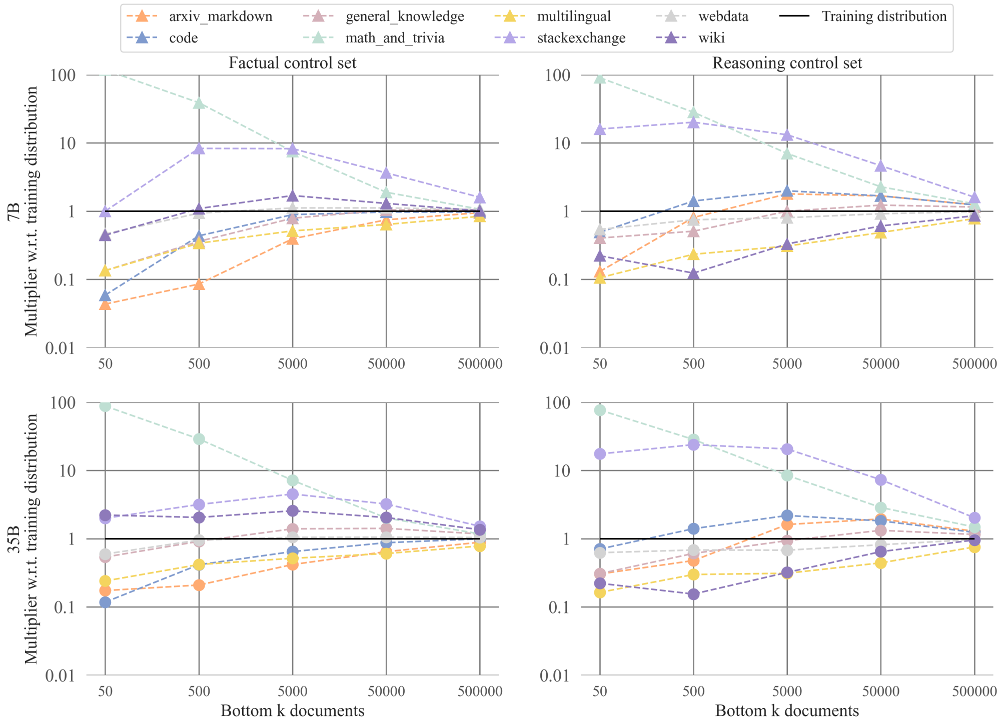


Figure B.6: For the query *control* sets, we also compare the amount of documents from a certain source dataset that show up in the *bottom* portions of the rankings to the amount you would expect to show up if you randomly sample from the pre-training distribution (indicated by 'Training distribution' in the figure). The top two plots are for the 7B, and the bottom for the 35B. We find that it again looks similar to the source distribution for the top of the rankings for the query control sets.

B.8.5 Content analysis of relevant documents

We provide further insights into the characteristics of influential documents on reasoning queries. To do so, we compute capability categories of the $n = 500$ most frequently occurring documents among the $k = 5000$ most (top) or least (bottom) influential documents for the reasoning queries (for the 7B model), and compare these to a randomly sampled set of 500 documents (we repeat the sampling process three times and provide mean and standard deviation scores on the detected capabilities). Results are shown in Figure B.7. We can see that the "code" category represents the vast majority of most and least influential documents, whereas for the random subsets the fraction of code-related documents is relatively small. This provides further evidence that code-related documents strongly influence model performance on reasoning tasks.

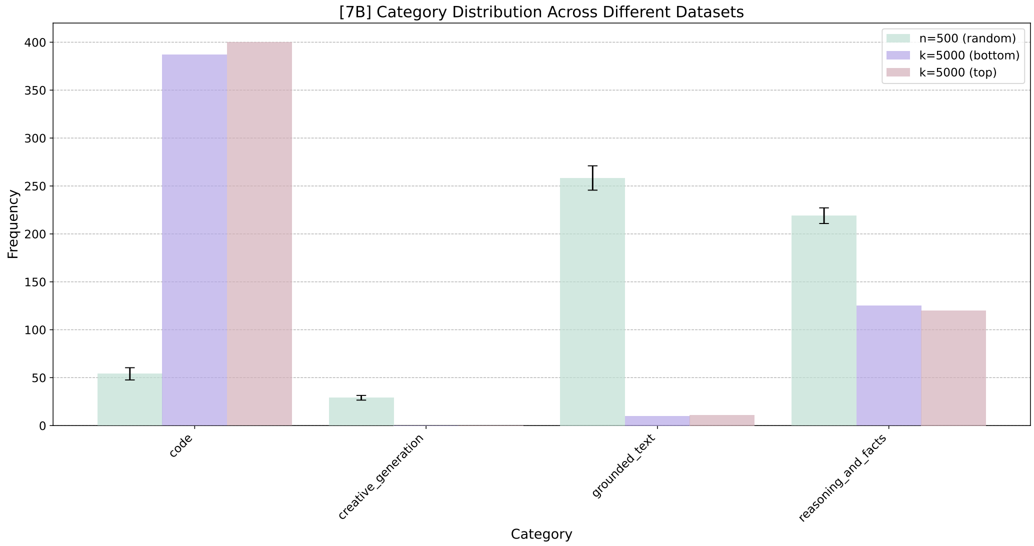


Figure B.7: Comparison of capability categories identified for the most and least influential documents for the reasoning queries, as well as for a random subset of sampled documents. We repeat the random sampling three times and report mean scores with standard deviations indicated.

B.9 Additional quantitative analysis

B.9.1 Correlation analysis

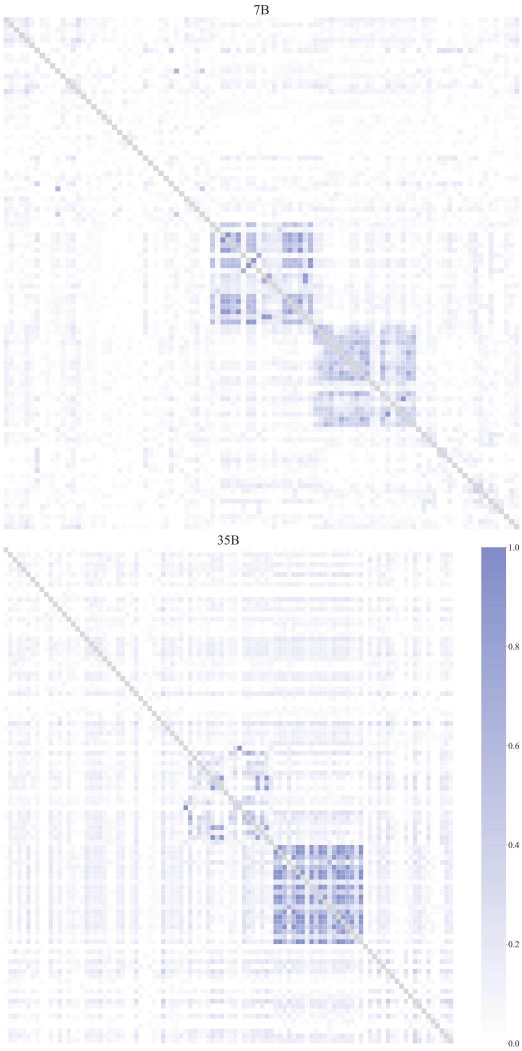


Figure B.8: The correlation between the influence scores of all 5 million documents for pairs of queries. All queries are on the x- and y-axis, with the first 40 belonging to the factual set, the next 40 to the reasoning set (arithmetic and slopes for the 7B, and linear and slopes for the 35B), the following 10 to the factual control set, and the last 10 to the reasoning control set. The take-away is that there is only a signficant correlation between queries of the same reasoning type, most strongly so for the 35B slopes queries.

Finding 1: correlation between reasoning queries of the same type.

In Chapter 3, we find that there is a correlation between the influence scores for the documents for different queries that underlie the same type of reasoning question (e.g. questions that all require calculating the slope but for different numbers). Recall that the correlation of influence scores for documents says something about

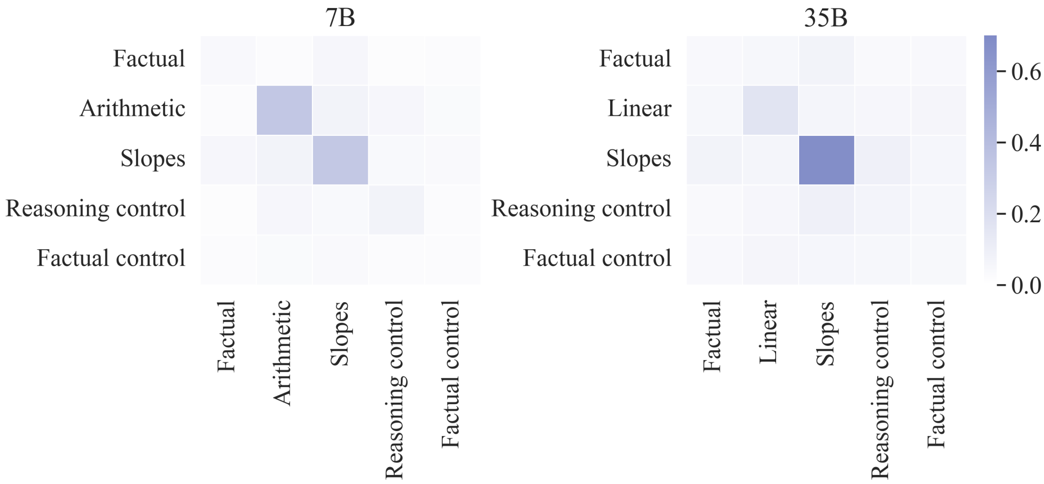


Figure B.9: The average correlations between the influences of all documents for queries of a specific type grouped. We leave out any query combinations where the correlation is not significant and any combination where the query on the x- and y-axis is the same query. We again observe that there is only a correlation of influence for queries of the same reasoning type.

whether or not models are learning a similar amount from the same data for their completions. Intuitively, correlations could be driven by many things, like formatting of the training documents, factual information it contains that is useful or necessary for the completions, implicit information about how to reason, explicit information about how to reason, N-gram overlap, etc. In the main text in Chapter 3, we take the higher correlations between reasoning queries of the same type to mean the models are generalising from a similar set of data to different reasoning questions. One other explanation for this result could be the fact that all these queries are superficially more similar to each other than the factual questions, and that this is the reason the influence correlates. To test this hypothesis, we use the 10 control queries for both the factual questions and the reasoning questions that are superficially similar, but do not require factual retrieval or reasoning to get to the solution (see Appendix B.3 for examples from each set). Besides testing this alternative hypothesis, this section gives more insight into what drives influence scores in LLMs.

Figure B.8 shows all query-query correlations for the 100 queries we look at, ordered as follows: 40 factual queries, 40 reasoning (of which 20 are arithmetic and 20 slopes questions for the 7B, and 20 linear equation and 20 slopes questions for the 35B), 10 factual control, and 10 reasoning control queries. We see that there is only a significant correlation between queries of the same reasoning type, which we summarise in Figure B.9 by grouping queries of the same type and averaging the correlations.

To get a sense of what the control questions look like, we show a comparison of a slopes query (top) and the control query that is meant to resemble it (bottom) below. The influences for this query correlate with the influences for the slope questions

Table B.16: Two query examples, of which one is from the reasoning set (slopes) and the other from the reasoning control set that resembles the slope queries superficially, but does not require any reasoning. The completions are by the 7B model.

Reasoning and Reasoning Control Query Set (7B)
Slopes (original prompt)
What is the slope of the line passing through the points (79, 47) and (80, 25)? Think step-by-step.
Completion
To find the slope of the line passing through the points (79, 47) and (80, 25), we can use the following steps: 1. Identify the coordinates of the two points: - Point A: (79, 47) - Point B: (80, 25) 2. Calculate the difference in x-coordinates between the two points: - Point A x-coordinate: 79 - Point B x-coordinate: 80 ... *redacted completion.*
Slopes (control prompt)
The slope of the line is -22. What is the slope of the line? Think step-by-step.
Completion
Given that the slope of the line is -22. To find the slope of the line, we need to find the change in y-coordinates divided by the change in x-coordinates. The slope of the line is -22. Therefore, the slope of the line is -22.

on average with a Pearson's R of 0.05, which is much smaller than the 0.32 average correlation between the influences found for the different slope reasoning questions by the 7B model. Below, we perform a more detailed qualitative analysis of the query combinations and what drives their correlations, but first we discuss the quantitative result.

As mentioned, we have 10 factual and 10 reasoning control questions for both models, and show the full correlation matrices below in Figure B.8 (per query) and Figure B.9 (averaged per group). We observe that the correlations between queries from the control sets and other query sets for the 35B is always between 0.05 and 0.10, which indicates that there can be a score correlation of at least 0.10 for other things than genuine reasoning (e.g. formatting, or topic). Further, the within-group correlations of the reasoning control set sometimes go as high as 0.38 (although the average is 0.06 for the 7B and 0.10 for the 35B). For comparison, the average linear-linear (subset of reasoning questions) score correlation for the 35B is 0.16, and not many of the correlations that make up this average are higher than the correlations in the reasoning control sets. To get a sense of how different the correlations are in magnitude between the reasoning questions and the control questions, we calculate the highest correlation of a query from a specific reasoning type with any other query that does not concern reasoning, and count the amount of reasoning query-query combinations for which the correlation is higher. For example, the maximum correlation we find between any slope question for the 35B and any other query that is not a slope question is 0.30 Pearson's R. If we discard all slope query combinations that are below 0.30 we are left with 138 of 190 significant combinations that are higher, ranging up to 0.96 Pearson's R (note that each reasoning group has 20 queries, and all combinations are $20 * 19/2 = 190$). For the linear equation queries by contrast, there are only 34 of 190 query-query combinations within this group that have a correlation higher than the highest correlation with the control queries, ranging up to 0.95 Pearson's R. For the 7B, 84 of 190 arithmetic query combinations have a higher correlation than the control correlations, ranging up to 0.96 Pearson's R, and 120 of 190 slopes query combinations, ranging up to 0.88. We therefore conclude that the correlations between the queries for the linear equations can mainly be explained by other, more superficial things than procedural knowledge, and connect this finding to the fact that the model is less robustly able to solve linear equations. The within-group correlations of the factual set are much lower, and for the 7B we only find 5 of 780 correlations that are higher than the maximum correlation of a factual query with another query group, ranging to 0.63 Pearson's R (we show the queries with the highest correlation below). For the 35B, we find no correlations for factual queries higher than the maximum correlation with another group.

We release all $100 * 100$ query-query correlations in the supplemental material[5] (on which Figure B.8 is based), and we highlight a few examples here to get a sense of what is driving higher or lower correlations. We mostly do not find a correlation between the influence scores for different factual questions (on average the correlation is 0.06 for the 7B and 0.03 for the 35B), but we show the highest correlation found between two factual queries below in Table B.17. For this particular example, the correlation seems driven by the fact that they are both "what"-questions with very short completions using quotation marks. By contrast, a similar style query-completion pair for the 35B shows almost no correlation (Table B.18). For the factual control set (the average correlation is 0.04 for the 7B and 0.08 for the 35B, but we also show a factual control correlation below for completeness. Table B.19 shows a factual query and a matched factual control example for the 35B model. The correlation between this control query and the factual query it matches is -0.03.

Table B.17: Two query examples for which the Pearson's R correlation of their document influences is 0.63, both from the factual set of the 7B model. We observe that the main similarity between these queries is that they both concern short "what" questions, and the completions both are short answers with quotes used.

Factual Query Set (7B)
Factual prompt 1
What is the symbol for gold on the periodic table?
Completion to factual prompt 1
The symbol for gold on the periodic table is "Au".
Factual prompt 2
What is the common name for the larva of a housefly?
Completion to factual prompt 2
The common name for the larva of a housefly is "maggot"

See Table B.20 below for a representative high correlation example for the 35B slopes questions (Pearson's R of 0.89):

We see that the numbers for the questions are different, but the steps to get to the answer are the same and formatted similarly. We generally observe that a combination of different numbers used in the same procedure and similarly formatted reasoning steps results in higher correlation. Another example is shown in Table B.21 below for a query combination that gets a lower correlation (0.55), but has

[5] https://openreview.net/forum?id=1hQKHHUsMx

Table B.18: Two query examples for which the Pearson's R correlation of their document influences is 0.08, both from the factual set of the 35B model. We observe that the main similarity between these queries is that they both concern short "what" questions, and the completions both are relatively short answers with quotes used.

Factual Query Set (35B)
Factual prompt 1
What is the name of the main antagonist in the Shakespeare play "Othello"?
Completion to factual prompt 1
The main antagonist in Shakespeare's play "Othello" is Iago, Othello's standard-bearer. Iago is a manipulative and deceitful character who orchestrates the tragedy in the play.
Factual prompt 2
What is the name of the smallest member of the camel family?
Completion to factual prompt 2
The smallest member of the camel family is called a "dromedary". Dromedaries are easily recognizable by their single hump. They are native to North Africa and the Middle East.

the same answer (-22). The relatively lower correlation here seems driven by the different formatting of the completion (less newlines). This indicates that part of the correlations are driven by superficial similarities such as completion formatting.

For comparison, Table B.22 shows a pair of control queries for which the prompt and completion are also very similarly formatted, and the correlation is 0.38 (which is the highest correlation we find using a control query).

However, superficial formatting cannot be the full picture, as we show in Table B.23, where the correlation is 0.35 but the completions are very dissimilar. For this particular query, it seems that it's not the positive scores that explain the correlation (Pearson's R of about 0.2), but rather the negative scores (Pearson's R of 0.44). Perhaps some of the correlation can be explained by documents that are generally negatively influential for the types of completions the models generate here.

Finally, in Table B.24 and B.25 we show two relatively high correlation examples for the arithmetic queries, again indicating that both underlying operations as well as formatting drive correlations.

Table B.19: Two query examples for which the Pearson's R correlation of their document influences is -0.03, one from the factual set (top), and one from the factual control set (bottom). We observe that although the formatting is similar, and there are some overlapping concepts, there is no correlation between the document influences.

Factual and Control Query Set (35B)
Factual prompt
What is the supernova remnant and pulsar wind nebula in the constellation of Taurus called?
Completion to factual prompt
The supernova remnant and pulsar wind nebula in the constellation Taurus is called the Crab Nebula.
Factual control prompt
In a fictional planetary system, there is a supernova remnant and pulsar wind nebula in the centre of a constellation of stars called Minotaur called the Goat Nebula. What is the supernova remnant and pulsar wind nebula in the constellation Minotaur called?
Completion to factual control prompt
In the fictional planetary system, the supernova remnant and pulsar wind nebula in the centre of the constellation Minotaur is called the "Goat Nebula".

Taken together, it seems like correlations can be driven by underlying procedures, formatting of the completion, and other more general things (like "what"-questions in Table B.17 and B.23). We generally find the highest correlations when procedures and formatting of completions coincide (of which two examples are given in Table B.20 and B.25). The magnitude of these correlations indicate that almost all of the influence of the 5 million documents in similar for such queries. One interesting possibility is that the query information surrounding the actual numbers generated (which do not seem to drive correlation much at all) is determined by the attention layers (which, besides the dense parameters contained in them, we ignore in this thesis), connecting potentially to literature attributing reasoning operations to attention heads. An interesting avenue for future work would be investigating this further.

7B vs 35B

An additional finding that is not central to the research question in Chapter 3, but is nonetheless interesting, is that there is almost no correlation between the influence scores of the two different models. We have 36 queries that share the same prompt for the 7B and 35B (16 factual questions, and 20 slopes reasoning questions) and we can calculate the Pearson's R of the queries with matched prompts (i.e. 36 combinations). The average correlation of influence scores is 0.02 Pearson's R (if we only look at the slopes questions the average correlation is 0.03). The maximum correlation we find is 0.19, for the question *"What is the capital of Belgium?"*, which we know from above is not a comparatively high score correlation. Interestingly, for this query, both models produced the exact same completion, and still the correlation is comparatively low. All other query combinations correlate with a Pearson's R below 0.11. This connects to a finding from Grosse et al. [Gro+23] (larger models rely on data that is more abstractly related to the prompt): the 35B model relies on very different pre-training data than the 7B, and the same pre-training documents influence completions for the same prompt very differently.

Table B.20: Two query examples for which the Pearson's R correlation of their document influences is 0.89, both from the reasoning query set concerning calculating slopes. We observe that, although the reasoning is applied to different numbers, the steps follow an identical pattern. In fact, the only difference between the two queries, including completions, is the numbers used.

Reasoning Query Set (35B)
Slopes (prompt 1)
What is the slope of the line passing through the points (94, 62) and (95, 19)? Think step-by-step.
Completion to prompt 1
The slope of a line passing through two points is calculated using the formula: Slope = (change in y) / (change in x) In this case, we have the points (94, 62) and (95, 19). Substituting the values into the formula, we have: Slope = (19 - 62) / (95 - 94) Slope = (-43) / (1) Slope = -43 Therefore, the slope of the line passing through the points (94, 62) and (95, 19) is -43.
Slopes (prompt 2)
What is the slope of the line passing through the points (90, 20) and (94, 64)? Think step-by-step.
Completion to prompt 2
The slope of a line passing through two points is calculated using the formula: Slope = (change in y) / (change in x) In this case, we have the points (90, 20) and (94, 64). Substituting the values into the formula, we have: Slope = (64 - 20) / (94 - 90) Slope = (44 / 4) Slope = 11 Therefore, the slope of the line passing through the points (90, 20) and (94, 64) is 11

Table B.21: Two query examples for which the Pearson's R correlation of their document influences is 0.55, both from the reasoning query set concerning calculating the slope. We observe that the second completion is structured very similarly to the first, except it has less new lines.

Reasoning Query Set (7B)

Slopes (prompt 1)

What is the slope of the line passing through the points (62, 31) and (61, 53)?

Think step-by-step.

Completion

The slope of a line passing through two points is calculated using the formula:

Slope = (change in y) / (change in x)

In this case, we have the points (62, 31) and (61, 53).

Substituting the values into the formula, we have:

Slope = (31 - 53) / (62 - 61)

Slope = (-22) / (1)

Slope = -22

Therefore, the slope of the line passing through the points
(62, 31) and (61, 53) is -22

Slopes (prompt 2)

What is the slope of the line passing through the points (79, 47) and (80, 25)?

Think step-by-step.

Completion

The slope of a line passing through two points is calculated using the formula:
Slope = (change in y) / (change in x)

In this case, we have the points (79, 47) and (80, 25).

Substituting the values into the formula, we have:
Slope = (25 - 47) / (80 - 79)
Slope = (-22) / (1)
Slope = -22

Therefore, the slope of the line passing through the points
(79, 47) and (80, 25) is -22.

Table B.22: Two query examples for which the Pearson's R correlation of their document influences is 0.38, both from the reasoning control set for the 7B model. We observe that the formatting is very similar, but the correlation is still lower than for the reasoning queries above.

Reasoning Control Query Set (7B)
Reasoning Control (prompt 1)
Find the answer: There are 21 kids in the bus. How many kids are in the bus? Think step-by-step.
Completion
There are 21 kids in the bus.
Reasoning Control (prompt 2)
Find the answer: There are 132 marbles in a jar. How many marbles are in the jar? Think step-by-step.
Completion
There are 132 marbles in the jar.

Table B.23: Two query examples for which the Pearson's R correlation of their document influences is 0.35, both from the reasoning control set for the 35B model. We observe that their completions are very different, indicating that it can not only be superficial similarities that drive correlations.

Reasoning Control Query Set (35B)
Reasoning control (prompt 1)
This line has a slope 6. What is the slope of the aforementioned line? Think step-by-step.
Completion
The slope of a line is a measure of the steepness of the line. It is defined as the ratio of the change in the y-coordinate to the change in the x-coordinate. In this case, the line has a slope of 6. This means that for every 1 unit increase in the x-coordinate, the y-coordinate will increase by 6 units. It's important to note that the slope of a line is not limited to positive values and can be negative or zero depending on the orientation of the line
Reasoning control (prompt 2)
Find the answer: there are two bananas on a tree, how many bananas are on the tree? Think step-by-step.
Completion
There are two bananas on the tree.

Table B.24: Two query examples for which the Pearson's R correlation of their document influences is 0.49, both from the reasoning query set concerning two-step arithmetic. We observe that one of the operations required is overlapping (+ 6), and the completion is somewhat similarly formatted but not fully.

Reasoning Query Set (7B)
Arithmetic (prompt 1)
Calculate the answer: (5 + 6) * 10 Think step-by-step.
Completion
First, let's evaluate the expression inside the parentheses: 5 + 6 = 11 Now, let's multiply the result by 10: 11 * 10 = 110 Therefore, the answer is 110
Arithmetic (prompt 2)
Calculate the answer: (6 + 10) * 6 Think step-by-step.
Completion
First, let's evaluate the expression inside the parentheses: 6 + 10 = 16 Now, let's calculate the result of multiplying the expression inside the parentheses by 6: 16 * 6 = 96 Therefore, the answer to the expression is 96

Table B.25: Two query examples for which the Pearson's R correlation of their document influences is 0.94, both from the reasoning query set concerning two-step arithmetic. We observe that not only is the completion very similarly formatted, but one of the operations overlaps (multiplying something by 2).

Reasoning Query Set (7B)
Arithmetic (prompt 1)
Calculate the answer: (5 - 3) * 12 Think step-by-step.
Completion
First, let's evaluate the expression inside the parentheses: 5 - 3 = 2 Now, let's multiply the result by 12: 2 * 12 = 24 Therefore, the answer is 24
Arithmetic (prompt 2)
Calculate the answer: (4 - 2) * 15 Think step-by-step.
Completion
First, let's evaluate the expression inside the parentheses: 4 - 2 = 2 Now, let's multiply the result by 15: 2 * 15 = 30 Therefore, the answer is 30

B.9.2 Magnitude of influence

Finding 2: magnitude of influence is much lower and less volatile for reasoning questions.

In Section 3.4.1, we find that the influence of documents at the same rank for factual questions is much more volatile than for reasoning questions. We mention that one explanation for this might be that the queries for the 35B model are much more niche, and therefore the relevant documents much more infrequent. To test this hypothesis, we plot the same results for only the overlapping queries (those that are part of both query sets for the 7B and 35B) in Figure B.10. We find that the magnitude and variance is still larger for the 35B model than for the 7B model, indicating that the influence of influential documents for the factual and reasoning questions by the 35B can be much larger than for the 7B model. Further, in Figure B.11 we show that the results look similar for the negative portions of the ranking (where we flip the influence scores from negative to positive).

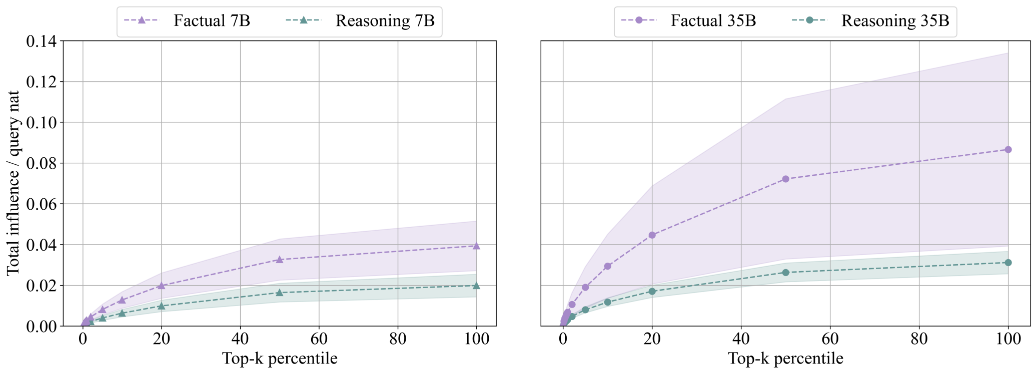


Figure B.10: The total influence per nat of query completion information for different portions of the *positive* ranking over documents, left for the 7B model, right for the 35B. In this case, we only plot queries that are present in the query sets for both models. This means the prompt is the same, but the completion is be different. The pattern is very similar as the observed pattern for the top of the ranking.

Finally, in Figure B.12 and Figure B.13 we plot the same metric for all queries for the top and bottom parts of the rankings respectively, now including the 10 control set queries of the factual and reasoning control set. As shown in Appendix B.3, we use 10 control queries for each set to investigate whether results hold similarly for queries that superficially look similar as the factual/reasoning questions, but that do not require factual retrieval or reasoning respectively. We observe that the control sets both show much higher variance and magnitude than the reasoning queries as well, for the positive and negative portions of the ranking. For completeness, we show the same result with the number of documents on the x-axis instead of percentiles in Figure B.14 and Figure B.15, to show that the results are similar if we

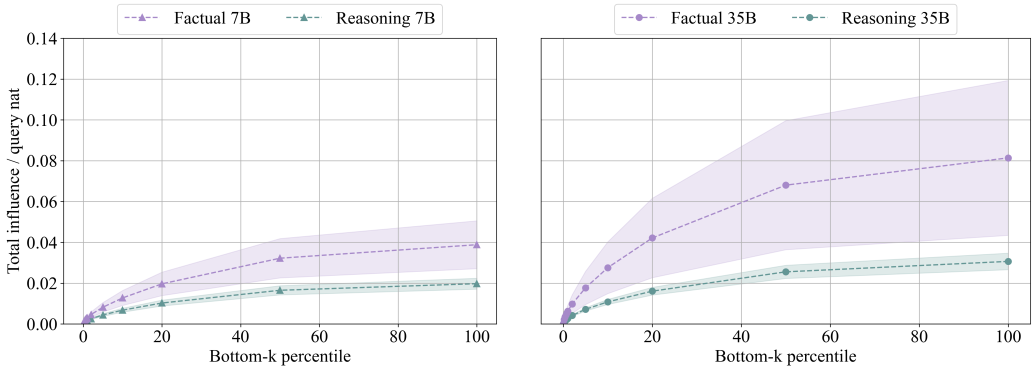


Figure B.11: The total influence per nat of query completion information for different portions of the *negative* ranking over documents, left for the 7B model, right for the 35B. We again only plot queries that are present in the query sets for both models. In this case, the k-th percentile contains the top k % of most negatively influential documents. The pattern is very similar as the observed pattern for the top of the ranking.

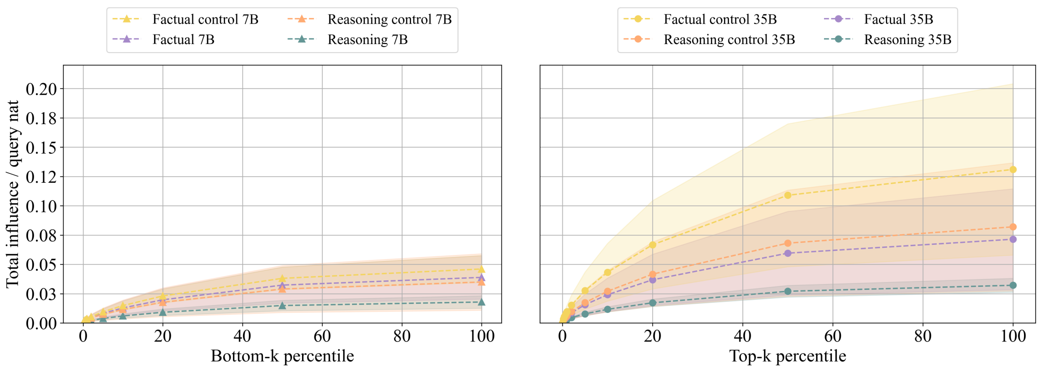


Figure B.12: The total influence per nat of query completion information for different portions of the *positive* ranking over documents, left for the 7B model, right for the 35B. We plot all queries, including the query control sets for both factual and reasoning, which contain 10 queries each.

take into account that the 20-th percentile of documents for each query contains a different amount of documents k.

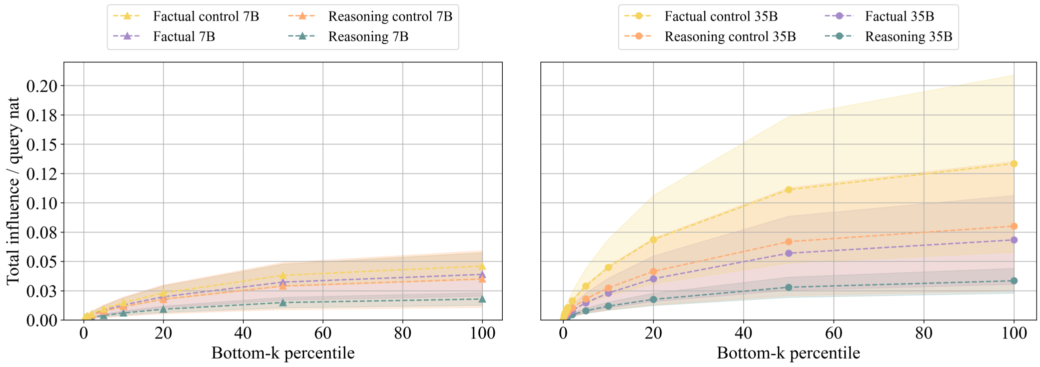


Figure B.13: The total influence per nat of query completion information for different portions of the *negative* ranking over documents, left for the 7B model, right for the 35B. We plot all queries, including the query control sets for both factual and reasoning, which contain 10 queries each.

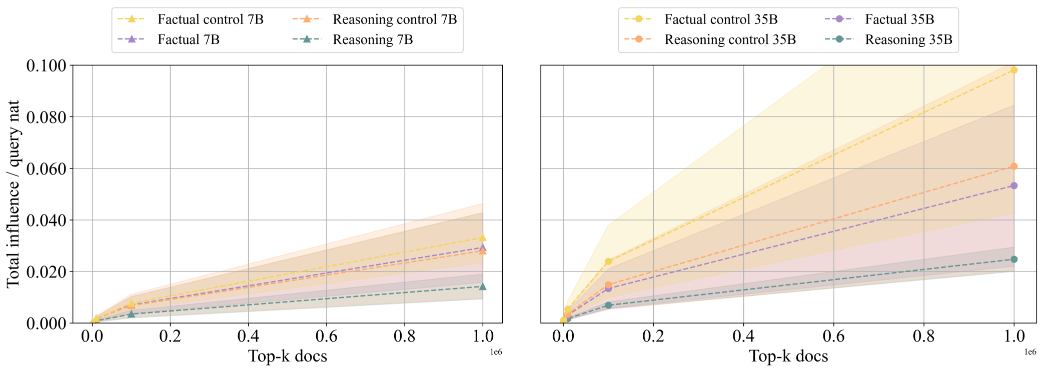


Figure B.14: The total influence per nat of query completion information for different number of documents k of the *positive* ranking, left for the 7B model, right for the 35B. We plot all queries, including the query control sets for both factual and reasoning, which contain 10 queries each.

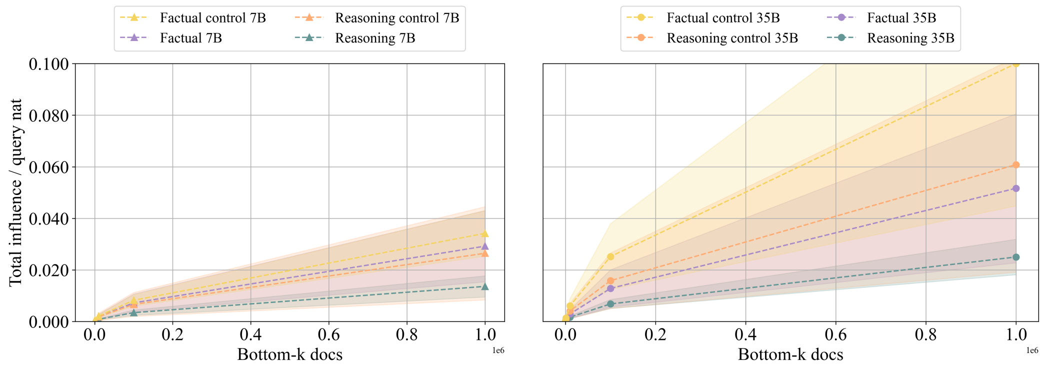


Figure B.15: The total influence per nat of query completion information for different number of documents k of the *negative* ranking, left for the 7B model, right for the 35B. We plot all queries, including the query control sets for both factual and reasoning, which contain 10 queries each.

B.9.3 Influence spread: power laws

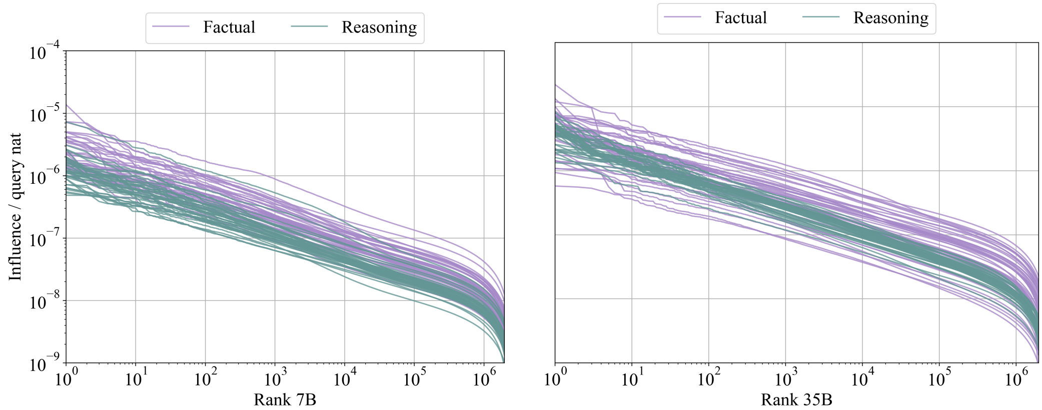


Figure B.16: The ranked influence scores per query nat for each query shown separately in log-log space. We observe; the results follow power laws (linear in log-log space), everything is shifted up for the 35B model (right), generally the scores for the reasoning documents are lower for the 7B model, and for the 35B model there is less variance in magnitude of influence for reasoning queries than for factual queries, and more often than not the influence scores are lower than for factual questions.

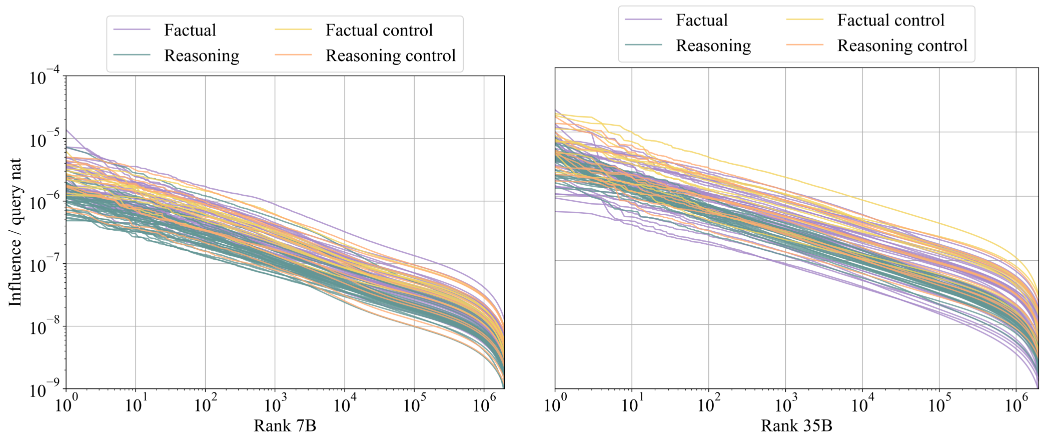


Figure B.17: The ranked influence scores per query nat for each query shown separately in log-log space again, but now also showing the control queries. We observe that also for the control queries the influence is much more volatile than for reasoning questions, and on average the magnitude is higher.

In this section, we look at the power laws induced by the top portions of the rankings. We can fit linear functions to the rankings in log-log space, and analyse the slopes to comment on the sparsity of the rankings (i.e. how many documents do models rely on for a completion). Specifically, we perform linear regression on the log-log top

500 rankings of each query, and report the slopes in Table B.26.

Table B.26: Slopes of the fitted functions to the top 500 documents in the influence rankings in log-log space, separated by query set and whether the model gets the question right or wrong. $\star$ indicates the significance of an independent T-test performed between the slopes of the factual vs. reasoning queries, where $\star$ indicates a p-value below 0.1 and $\star\star$ below 0.05.

	7B (Correct)	**7B (Incorrect)**	**35B (Correct)**	**35B (Incorrect)**
Reasoning ($\boldsymbol{\alpha}$)	$-0.36 \pm 0.03^{\star}$	-0.33 ± 0.02	$-0.36 \pm 0.04^{\star\star}$	$-0.38 \pm 0.04^{\star}$
Factual ($\boldsymbol{\alpha}$)	-0.34 ± 0.03	-0.34 ± 0.04	-0.32 ± 0.05	-0.34 ± 0.04

After qualitatively inspecting the queries for the 35B model with the steepest slope, we believe an explanation for this result may be 'noise' in the influence scores. For example, the query with the steepest slope ($\alpha = -0.45$) has as the most influential document a document that is seemingly entirely unrelated to the query. Namely, the query asks the question *"What is the slope of the line passing through the points (41, 23) and (18, 92)? Think step-by-step."*, and the top influential document is a snippet about the lunar eclipses and when and where they can be viewed which does not have high N-gram overlap with the query either:

> *December 8, 1946 — Total Lunar Eclipse — Rawaki, Phoenix Islands, Kiribati*
> *Max view in Rawaki*
> *Sunday, December 8, 1946 at 5:01 AM*
> *Global Type: Total Lunar Eclipse*
> *Rawaki: Partial Lunar Eclipse*
> *Began: Sun, Dec 8, 1946 at 3:13 AM*
> *Maximum: Sun, Dec 8, 1946 at 5:01 AM*
> *Ended: Sun, Dec 8, 1946 at 8:22 AM*
> *Duration: 5 hours, 10 minutes*
> *December 8, 1946 — Total Lunar Eclipse — Rawaki*
> *You are using an outdated browser, to view the animation please update or switch to a modern browser. Alternatively you can view the old animation by clicking here.*
> *Animation: How the Partial Lunar Eclipse Looked*
> *The total phase of this lunar eclipse was not visible in Rawaki, but it could be observed there as a partial lunar eclipse.*
> *More about the December 8, 1946 — Total Lunar Eclipse*

Phases and local times of this eclipse
Eclipses visible in Rawaki
All eclipses worldwide, from 1900 to 2100

This is the only query for which we observe an unrelated top 1 document, but for the 35B model we qualitatively observed seemingly irrelevant documents in the rankings more often (in the 7B we did not observe this). This connects to a finding from literature that for large models influence functions sometimes surface documents with high gradient norms that are unrelated to the query [BBD20; Gro+23; Cho+24]. As Grosse et al. [Gro+23] note, it is currently unclear whether this is true noise, or whether these are genuinely influential for the completions. Regardless, it seems like noise cannot easily explain the difference between the factual and slopes queries, as one would expect noise to show up equally everywhere.

Another way to visualise this result is to plot the percentage of total influence contained in different parts of the top ranking, which we do in Figure B.18 below. The results in this plot show that for the top-k percentile of most positively influential documents, the total percentage of positive influence is much higher than k (e.g. 20% of the total positive influence is contained in the top 5% of documents). Here, it is clear that on average, for the 35B model the total amount of influence contained in the top-k percentile increases faster for reasoning questions than for factual questions, indicating that a larger portion of the total positive influence is contained in the top portions of the rankings. In Figure B.19 we show the same result holds if we include the control queries. As Grosse et al. [Gro+23], it is not clear whether this is a sensible result to show because for each query we are dividing the total influence at each k by the sum of positive influence for that query (perhaps a large part of the positive influence gets cancelled out by negative influence), but we show the result here nonetheless for completeness. We know from the absolute results of the total influence at different portions of the ranking that each percentage of total influence at the top-k percentile a much lower value in absolute terms for reasoning than for the factual questions. If the relative result does not turn out to be noise, it is the case that of the total influence, a higher percentage is contained in the top portions of the rankings for reasoning questions than for factual questions. Taken together with the fact that the absolute influence is often much higher for factual questions, this indicates that the model relies on more highly influential documents for factual retrieval than for reasoning. This could indicate that there are more highly relevant factual documents further down the ranking, which makes sense given the fact that the pre-training distribution is dominated by websources and news, which are more likely to contain relevant information for factual question answering than for reasoning. Further, it connects to the finding from literature that models need to see examples often before text gets memorised [Cho+22].

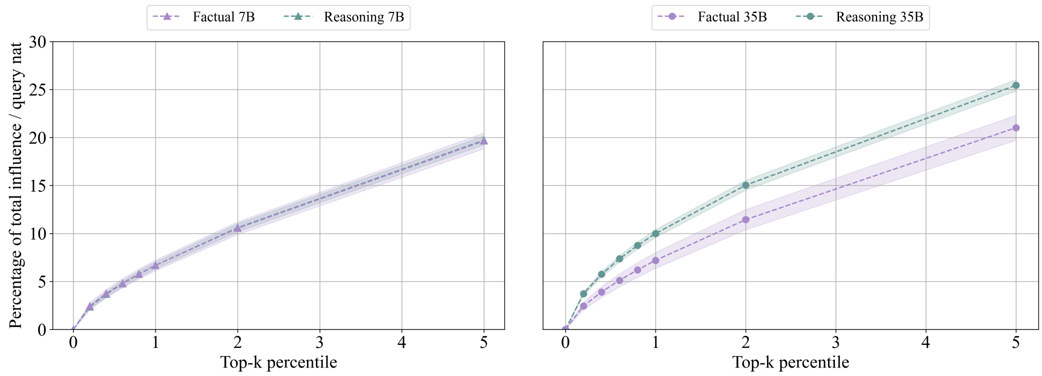


Figure B.18: The percentage of total influence per nat of query completion information for different portions of the *positive* ranking over documents, left for the 7B model, right for the 35B. We plot only non-control queries.

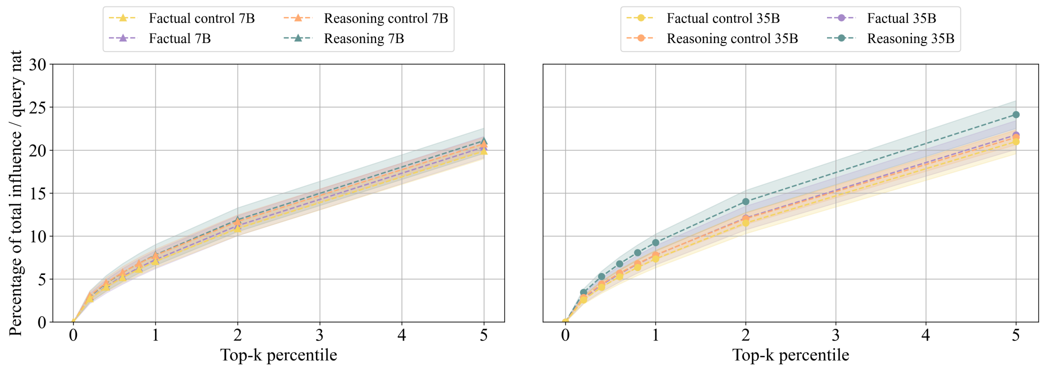


Figure B.19: The percentage of total influence per nat of query completion information for different portions of the *positive* ranking over documents, left for the 7B model, right for the 35B. We plot all queries, including the query control sets for both factual and reasoning, which contain 10 queries each.

Again, the picture looks similar for the negative portions of the ranking, shown for completeness below in Figure B.20 and B.21.

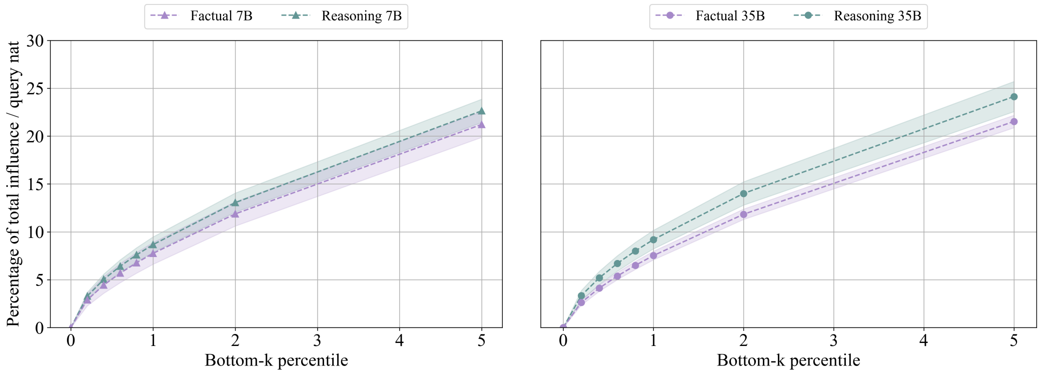


Figure B.20: The percentage of total influence per nat of query completion information for different portions of the *negative* ranking over documents, left for the 7B model, right for the 35B. We plot only non-control queries.

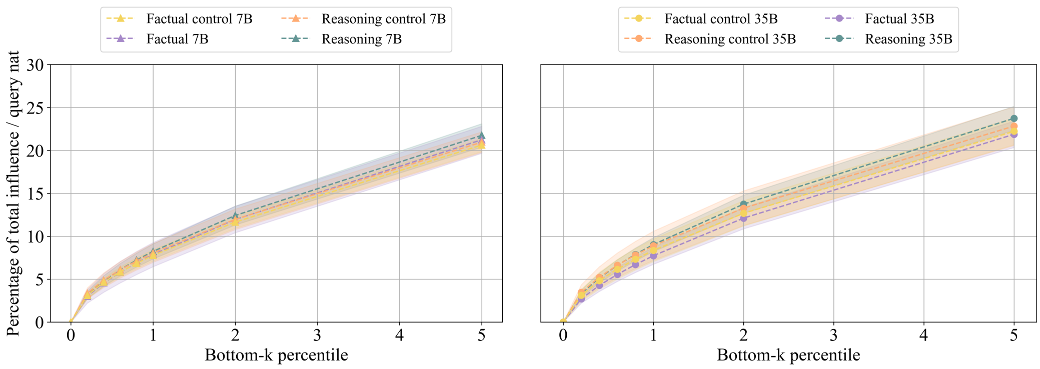


Figure B.21: The percentage of total influence per nat of query completion information for different portions of the *negative* ranking over documents, left for the 7B model, right for the 35B. We plot all queries, including the query control sets for both factual and reasoning, which contain 10 queries each.

Appendix C

How Models Learn to Reason from Code Data

This chapter contains the supplementary materials for Chapter 4. Below, I outline the contents of each section briefly.

Section C.1 contains the hyperparameters we used to fine-tune the models with SFT and GRPO.

Section C.2 presents additional results for larger dataset sizes.

Section C.3 ablates the two-stage nature of the experiments in Chapter 4, finding that two separate stages lead to higher sample efficiency.

Section C.4 discusses the finding that GRPO outperforms DPO and SFT when eliciting retroactive programming by backprop.

Section C.5 contains the distribution over inputs used in the cipher experiments in the main text at the end of Section 4.4.

Section C.6 has an example of a natural language description of a program.

Section C.7 details the compute used in the experiments.

C.1 Hyperparameters

C.1.1 SFT

All SFT experiments use a learning rate of 2×10^{-5} and a linear learning rate schedule. The Adam optimiser is used with $\beta = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. The max grad norm is set to 1.

C.1.2 RL

All RL experiments use GRPO with a group size of 6 batch size of 6. The learning rate is set to 1×10^{-6}. The Adam optimiser is used with $\beta = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$.

C.2 Data Scaling

C.2.1 Ablation over Dataset Size

Figure C.1 compares the performance of Llama models (1B, 3B and 8B parameters) for varying dataset size on the evaluation of Random Arithmetic programs. Here, 'dataset size', refers specifically to the amount of unique code functions included in the dataset. Performance is evaluated on three separate sets:

- The *w/ IO Train* set: both the function and the IO pairs are observed during training
- The *w/ IO Test* set: uses the same functions as *w/ IO Train* but different IO pairs, not included in the training data
- The *w/o IO Test* set: evaluates IO pairs for functions seen only as code during training

The results show that accuracy on both *w/ IO* and *w/o IO* sets generally increases with larger dataset sizes and larger model scales. Notably, model performance is strongly tied to parameter count; for example, the 8B model trained on only 100 unique functions achieves comparable performance on the *w/o IO* set to the 1B model trained on 800 functions.

C.2.2 Ablation over Number of IO Pairs

In Figure C.2 we vary the number of IO training pairs (per program) provided for the *w/ IO* set, and examine the results. This analysis specifically uses the `Llama-3.2-3B-Instruct` model on the Random Arithmetic dataset, which for this experiment consists of 200 distinct functions. Performance is reported across the same sets as the ones described in Appendix C.2.1. The results show how increasing the quantity of IO examples for each program affects not only direct generalisation in the *w/ IO* Test set, but also the model's ability to accurately execute *w/o IO* programs.

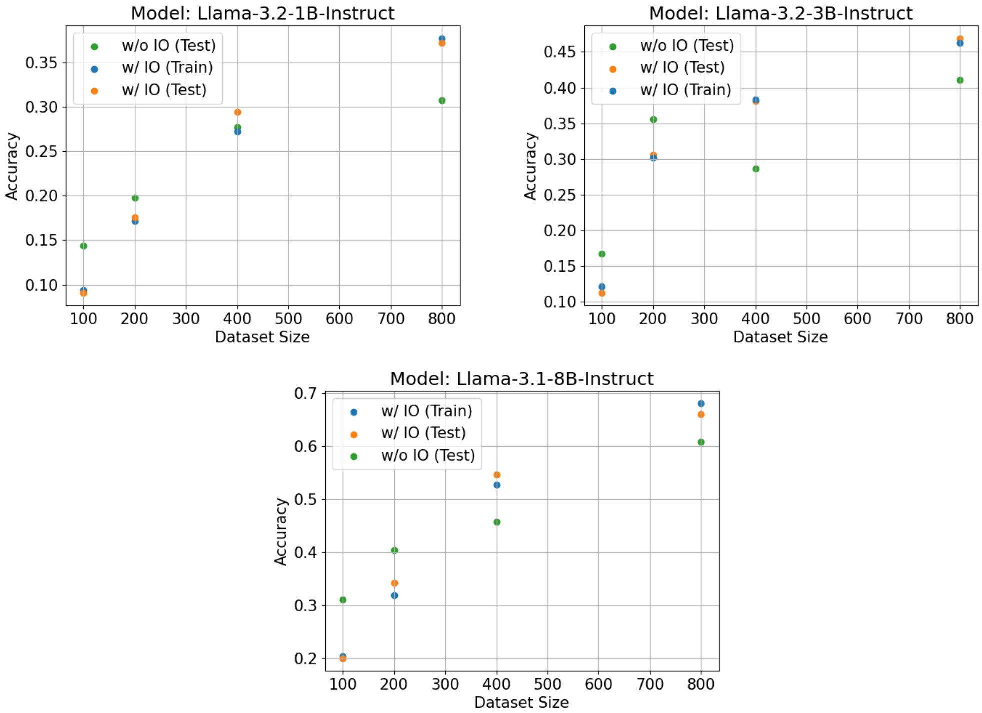


Figure C.1: Performance comparison of Llama models across 1B, 3B and 8B on *w/ IO* and *w/o IO* Random Arithmetic program evaluation. Each model is trained and tested across varying dataset sizes. Dataset size refers to the number of unique functions present in the dataset.

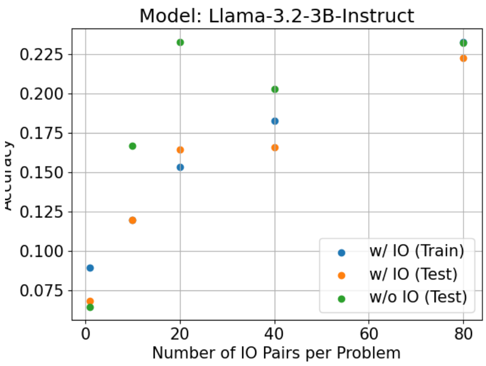


Figure C.2: Impact of varying the number of IO training pairs for *w/ IO* programs and *w/o IO* sets evaluation accuracy. Results are shown for the `Llama-3.2-3B-Instruct` model using a Random Arithmetic dataset comprising 200 distinct functions.

C.3 Single-Stage Programming by Backprop

In Figure C.3, we show the accuracy of Llama-3.1-8B-Instruct on *w/o IO* Random Arithmetic program evaluation following Proactive-PBB in comparison to a single SFT stage with all training data in a single mixture. As we scale the number of times the same piece of *w/o IO* source code appears in the dataset, with prompt and response preamble augmentations, single-stage SFT approaches the performance of Proactive-PBB. The greater sample efficiency of Proactive-PBB is likely because initial train steps on source code are waisted in single-stage SFT, as a code-I/O relationship has not yet been learned.

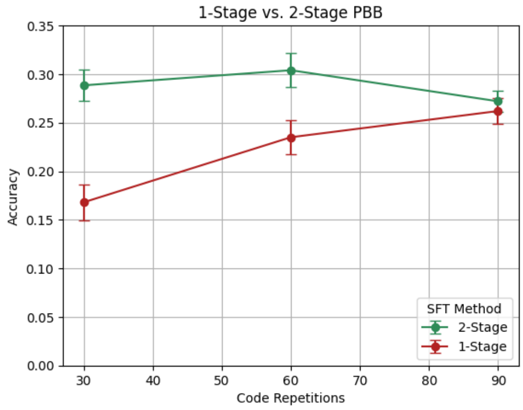


Figure C.3: Comparing two-stage Proactive-PBB to a single SFT stage on the full Random Arithmetic training data mixture for different numbers of repeated source code samples. The base model is Llama-3.1-8B-Instruct.

C.4 Online vs. Offline Retroactive-PBB

In Figure C.4, we compare different fine-tuning algorithms for the second stage of Retroactive-PBB with Llama-3.1-8B-Instruct on Random Arithmetic. DPO allows for learning from both positive and negative samples, considerably outperforming SFT. GRPO is an online RL algorithm, meaning that the model learns from on-policy data, which could be why it yields further improvements. To ensure a fair comparison, we use only 15 I/O pairs instead of 90 $(90/6 = 15)$ per *w/ IO* program for RL training. With a group size of 6, this means that for each I/O pair the model generates 6 completions that receive a reward. This in turn leads to the same number of completions with a ground-truth signal as SFT, and a fairer comparison.

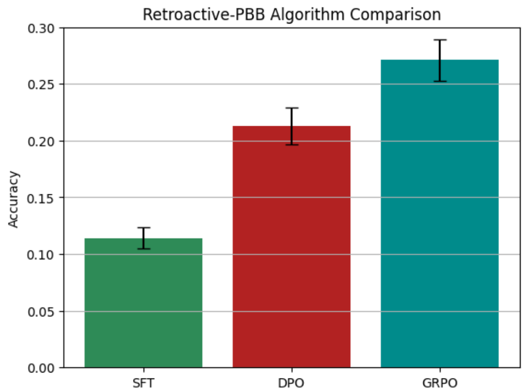


Figure C.4: Comparing fine-tuning algorithms for the second stage of Retroactive-PBB on Random Arithmetic with Llama-3.1-8B-Instruct. DPO is an offline method, but allows for learning from positive and negative examples. GRPO is online and thus has the added benefit of learning from on-policy data.

C.5 Ciphers Data

A plot showing the distribution of IO pairs used in Figure 4.4 is provided in Figure C.5.

C.6 Natural Language Descriptions

Here, we include an example of a random arithmetic program and its natural language description.

Program:

```
def Blaankle(x):
    t0 = x + x
    t1 = 1 * abs(t0)
    return t1
```

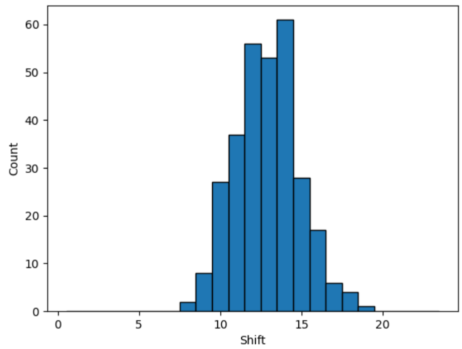


Figure C.5: Sampled shifts for cipher I/O pairs.

Description: *A Blaankle is a process that takes an input value, doubles it, and then returns the absolute value of the doubled result.*

C.7 Compute Requirements

All experiments with `Llama` models can be run on two GPUs with 40GB vRAM. We used data parallelism over 4 NVIDIA L40s GPUs to run these experiments. Experiments with `GPT-4o` made use of the OpenAI fine-tuning API. Data generation (Leetcode word problems and post-rationalised chain-of-thought ground truth outputs for all datasets) and fine-tuning runs came to a total cost just over 500 USD.

Appendix D

A Case Study in Social Reasoning: Pragmatics

This Chapter contains the appendix for Chapter 5. Below, I briefly outline the content of each section for reference.
I begin with a more detailed background on conversational implicature in Section D.1.
Then, I provide all the prompt templates used for the evaluation in Section D.2.
A categorisation of models evaluated is given in Section D.3.
Details on the human experiments can be found in Section D.4.
A comment on the fact that BIG bench uses the same underlying benchmark in Section D.5.
Examples of completions by the model with chains-of-thought in Section D.6.
A detailed table of results for all experiments can be found in Section D.7.
Section D.7 further contains the following additional experiments:

- We reframe the task as a contrastive one in Section D.7.1, finding it works less well.
- We report the variance over prompt ordering in Section D.7.2.
- We experiment with more detailed zero-shot prompts in Section D.7.3.
- We present detailed results for different prompt templates in Section D.7.4.
- We report variance of multiple API calls in Section D.7.5, finding it is minor.
- We confirm that few-shot in-context labels can be randomised without loss of performance in Section D.7.6.
- We do chain-of-thought prompting on base models in Section D.7.7.

- We test for spurious correlations and memorisation in Section D.7.8.

For reproducibility, we provide timestamps for all experiments done with APIs in Section D.8.
Finally, in Section D.9 we detail the compute used.

D.1 Background on implicature

The first influential consideration of implicature is [Gri75]. In his work, Grice continues the trend of moving away from purely logical accounts of language started by [Wit21] by hypothesising implicatures arise in conversation when some mutually agreed upon maxims seem to be violated. For example, if we agree on only making relevant contributions to conversation, Juan's response in the introduction seemingly violates this maxim—after all, he starts talking about work when Esther asks him about a party. However, because Juan agreed to be relevant he must be implying that having to work means he cannot come to the party. Grice contrasts conversational implicatures that arise through context with conventional implicatures. These are implicatures where the *conventional* meaning of the word determines what is implicated. An example given by Grice is the following sentence: "he is an Englishman; he is therefore brave.". Grice notes that this sentence does not literally state that an Englishman being brave is a direct consequence of him being English, but it's implied by the conventional meaning of the word 'therefore'.

Since then, issues with the Gricean cooperative principle have been pointed out by many [Lev83; SW86; Dav98; LS14]. The most influential alternative theory is relevancy theory by [SW86]. They do away with the cooperative principle and instead theorise implicatures arise because speakers try to produce utterances that are both as relevant as possible and require the least effort to process. Another point of contention is the incorporation of conventional implicatures on the pragmatics side. [Bac99] argues that there is no such thing as conventional implicatures, and they are simply instances of something else. Based on a thorough treatment of what Grice calls conventional implicatures, Bach argues all examples of it can be filed under other concepts within semantics, like utterance modifiers (called "utterance modifiers" instead of "sentence modifiers" because they go against the semantic content of the rest of the sentence). [Pot05] also argues that to explain conventional implicatures we can stay on semantic turf. Indeed, even Grice himself says conventional implicatures derive from the meaning of the words, not from conversational context. However, Potts does not claim conventional implicatures do not exist, but instead argues they arise by a combination of lexical meaning and novel ways of combining words—the latter being the well-known principle of compositionality, an important part of semantics, not of pragmatics. Potts provides us with an illuminating demarcation between conventional and conversational implicatures. Conventional implicatures are never negotiable by context, whereas conversational implicatures are context-dependent and can always be cancelled without causing incoherent discourse. Consider again the sentence "he is an Englishman; he is therefore brave." and the sentence "Eddie has three bicycles" (implicating that Eddie has exactly three bicycles and not more). The former sentence can not be cancelled by new context without contradiction, whereas for the latter, if we continue saying "In fact, Eddie has 10 bicycles, he is a bicycle junkie", we have cancelled the implicature. This demarcation clearly puts conventional implicatures on the semantic side, and conversational implicatures on the pragmatic side. Potts goes on by providing a formal theory for conventional implicatures.

In later work, [Pot06] describes how pragmatic pressures interacting with context cause conversational implicature to arise. He shows how sensitive conversational implicatures are to small changes in the context. Novel information about a speaker's belief state might completely change what is implied. There are many more models of implicature that aim to explain how humans understand language in context. Most notably, [FG12] formalise the view that speakers produce utterances that are helpful and not longer than necessary with a Bayesian model called the rational speech act (RSA). Many variants on the RSA framework have since been proposed. For example, [GF16] extend it to handle nonliteral uses of language, like irony, and metaphor. In the context of computational models, prior work uses insights from pragmatics to show that the use of certain words can make a language model produce biased completions ([PP21], e.g. saying someone "claimed" something rather than "said" something), and inform bias and sentiment classifiers [GR09; RDJ13].

In Chapter 5, we focus on conversational implicatures and not on conventional implicatures. All conversational implicatures are negotiable by context, but the way they depend on context can be different. [Gri75] identifies generalised conversational implicatures and particularised conversational implicatures. The former require little or no context to be resolved. For example, "some athletes smoke" can imply "not all athletes smoke", but might also imply "I do not know whether all athletes smoke" when it is a response to the question "do you know whether all athletes smoke?" [Dav19]. The latter only arise in certain contexts. For example, the response "I have an early morning" to the question "do you want to stay for a drink?".

D.2 Detailed prompt templates

Table D.1 contains the full prompt templates we used for the main evaluation and Table D.3 contains the extra prompt templates.

Table D.1: *Ranking prompt templates.* The six templates we wrap the test examples in to present to the models. Template 1, 3, and 4 are of the type *structured*, and 2, 5, and 6 of the type *natural.* Within the type of prompt template they only differ slightly in wording.

#	Prompt template
1	Does the following response to the question imply yes or no? question: <*utterance*> response: <*response*> implicature: <*implicature*>
2	Finish the following text: Esther asked "<*utterance*>" and Juan responded "<*response*>", which means <*implicature*>
3	Is the implied meaning of the following response yes or no: question: <*utterance*> response: <*response*> meaning: <*implicature*>
4	What is the intent of the following response, yes or no? question: <*utterance*> response: <*response*> intent: <*implicature*>
5	Finish the following text: Karen asked "<*utterance*>" and William responded "<*response*>", which means <*implicature*>
6	Finish the following text: Bob asked "<*utterance*>" and Alice responded "<*response*>", which means <*implicature*>

Table D.2: *Completion prompt templates.* The six adjusted templates we wrap the test examples in to present to the models when we use completion instead of likelihood ranking. Template 1, 3, and 4 are of the type *structured*, and 2, 5, and 6 of the type *natural*. Within the type of prompt template they only differ slightly in wording.

#	**Prompt template**
1	Does the following response to the question imply yes or no? Only output 'yes' or 'no'. Even if you're uncertain, choose either 'yes' or 'no'. question: <*utterance*> response: <*response*> implicature: <*implicature*>
2	Finish the following text. Only output 'yes' or 'no'. Even if you're uncertain, choose either 'yes' or 'no'. Esther asked "<*utterance*>" and Juan responded "<*response*>", which means <*implicature*>
3	Is the implied meaning of the following response yes or no. Only output 'yes' or 'no'. Even if you're uncertain, choose either 'yes' or 'no'. question: <*utterance*> response: <*response*> meaning: <*implicature*>
4	What is the intent of the following response, yes or no? Only output 'yes' or 'no'. Even if you're uncertain, choose either 'yes' or 'no'. question: <*utterance*> response: <*response*> intent: <*implicature*>
5	Finish the following text. Only output 'yes' or 'no'. Even if you're uncertain, choose either 'yes' or 'no'. Karen asked "<*utterance*>" and William responded "<*response*>", which means <*implicature*>
6	Finish the following text. Only output 'yes' or 'no'. Even if you're uncertain, choose either 'yes' or 'no'. Bob asked "<*utterance*>" and Alice responded "<*response*>", which means <*implicature*>

Table D.3: The three additional templates we wrap the test examples in to present to the models, adapted from [Gla+22].

#	**Prompt template**
7	The following text shows an interaction between two humans called Esther and Juan. In the interaction, Esther will ask Juan a question, and Juan will give an answer that contains an implicature. An implicature is an utterance that means something other than the literal meaning of the words. The implicature of Juan's response is yes or no. You, the AI assistant, are asked to finish the text with yes or no. The task begins: Esther asked "<*utterance*>" and Juan responded "<*response*>", which means <*implicature*>
8	The following text shows an interaction between two humans called Esther and Juan. In the interaction, Esther will ask Juan a question, and Juan will give an answer that has a meaning besides the literal meaning of the words. That meaning is either yes or no. You, the AI assistant, are asked to finish the text with the correct meaning, either yes or no. The task begins: Esther asked "<*utterance*>" and Juan responded "<*response*>", which means <*implicature*>
9	The following text shows an interaction between two humans called Esther and Juan. In the interaction, Esther will ask Juan a question, and Juan will give an answer that has a meaning besides the literal meaning of the words. That meaning is either yes or no. You, a highly intelligent and knowledgeable AI assistant, are asked to finish the text with the correct meaning, either yes or no. The task begins: Esther asked "<*utterance*>" and Juan responded "<*response*>", which means <*implicature*>

Table D.4: *Chain-of-thought (CoT) prompt templates.* One of the six chain-of-thought prompt templates we use for the CoT experiment. Note that this is a 5-shot prompt. Each prompt variation contains five CoT examples. The other five variations are separately added to the supplementary materials

#	**Prompt template**
1	Bob asks Alice a question, and Alice responds with an implicature. This means that Alice's response does not literally contain the answer to Bob's question, but implies an answer. Assuming that Alice is a cooperative conversational partner, what is the implicated answer to the question? For example: Bob: You invented fire? Alice: I told you that. Implicature: Alice says 'I told you that'. Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. Alice must mean that she told Bob that she invented fire. Alice's response to Bob's question 'You invented fire?' is yes. Answer: yes Bob: That cake looks delicious. Aren't you going to have some with me? Alice: But that was a huge meal we just had. Implicature: Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. Alice must mean that the meal they just had was so huge she is not hungry anymore, and this must be relevant to Bob's question: 'Aren't you going to have some with me?' Alice's response to the question must therefore be no. Answer: no Bob: Could you perform well? Alice: Being bilingual would help put me ahead of the pack. Implicature: Alice says being bilingual would help put her ahead of the pack. Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. Alice must be implying that she could perform well because she is bilingual. This means the response to Bob's question is yes. Answer: yes Bob: Have you any news for me? Alice: I've made progress Implicature: Alice says she has made progress. Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. If Alice would not have any news for Bob, Alice would not have said she would have made progress because that would be misleading. The answer to Bob's question 'Have you any news for me?' must therefore be yes. Answer: yes Bob: You looked out for him? Alice: He looked out for me. He taught me. Implicature: Bob asks Alice 'You looked out for him?' and Alice's response says that the person that is being referred to by 'him' here looked out for Alice. If Alice meant yes to Bob's question, Alice would have said something like 'he also looked out for me'. Stating the response like this implies that the answer to Bob's question is no. Answer: no Only output a 'yes' or 'no' as a final answer. Write your reasoning after 'Implicature:' and then output either 'Answer: yes' or 'Answer: no'. Bob: <*utterance*> Alice: <*response*> Implicature:

D.3 Model categorisation

Table D.5 contains details on the model classes that are a part of each group of models we evaluate, along with their model sizes.

Table D.5: Model categorisation for each of the models. DL stands for dialogue, FT for fine-tuning, BL for benchmark-level, EL for example-level, and IT for instruction-tuning.

Group	Model class	Model IDs	Model size	Instruct
Base	BERT	base uncased	110M	No
	RoBERTa	base, large	125M, 355M	No
	GPT-2	GPT-2 medium, large, xl	354M, 774M, 1.6B	No
	EleutherAI	GPT-J, GPT-NeoX	125M, 1.3B, 2.7B, 6B, 20B	No
	BLOOM	-	560M, 1B1, 3B, 7B1, 176B	No
	OPT	-	125M, 350M, 1.3B, 13B, 30B, 66B, 175B	No
	Cohere	small, medium, large, XL	409.3M, 6.067B, 13.12B, 52.4B	No
	GPT-3	ada, babbage, curie, davinci	Est. 350M, 1.3B, 6.7B, 175B	No
DL FT	BlenderBot	-	90M, 2.7B, 9.4B	No
BL IT	T0	-	3B, 11B	Yes
	Flan-T5	-	780M, 3B, 11B	Yes
EL IT	Cohere-command	medium, xlarge	6.067B, 52.4B	Yes
	text-davinci-001	ada, babbage, curie, davinci-1	Unknown, left-to-right increasing in size	Yes
	text-davinci-002	-	Unknown	Yes
	text-davinci-003	-	Unknown	Yes
	ChatGPT	gpt-3.5.turbo	Unknown	Yes
	GPT-4	gpt-4	Unknown	Yes

D.4 Human evaluation

The participants for the human evaluation in Chapter 5 were recruited using Prolific (www.prolific.co). The setup of the experiment is as follows. We divide the test set of 600 examples into four non-overlapping subsets of 150 examples. Each set of 150 examples was given to five unique annotators. This means each example in the test set is labeled five times by different people, and we have in total twenty annotators for the whole test set (five different ones for each of the four subsets). The only constraint for the annotators is that they are native English speakers. In Figure D.1 the screen shown to potential participants on Prolific is shown. Participants are paid 15 pounds an hour, which was the living wage at the time of the experiment and more than the 12 dollars an hour Prolific recommends. The total amount spent on the human evaluation is 236 pounds. This amount came to be from four subsets, each costing about 30 minutes to label per annotators, and having 5 annotators per subset: 15 * 4 * 0.5 * 5 = 150. The extra costs were for the annotator that didn't pass the attention check which we paid nonetheless, and for prolific as a platform.

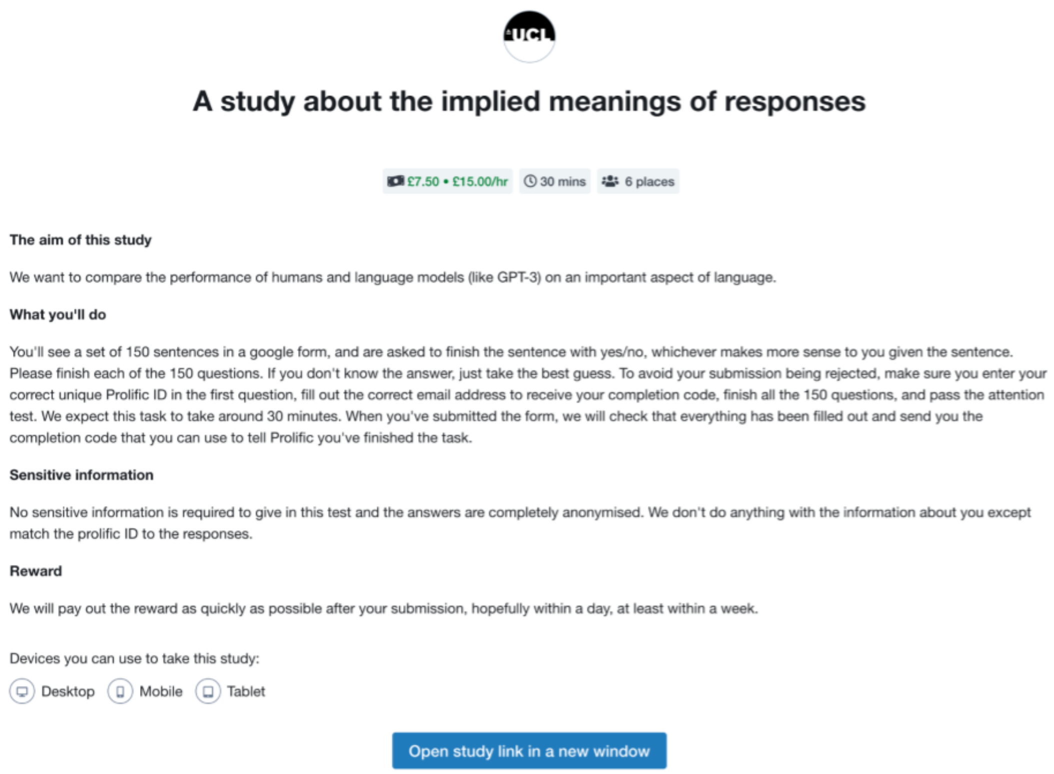
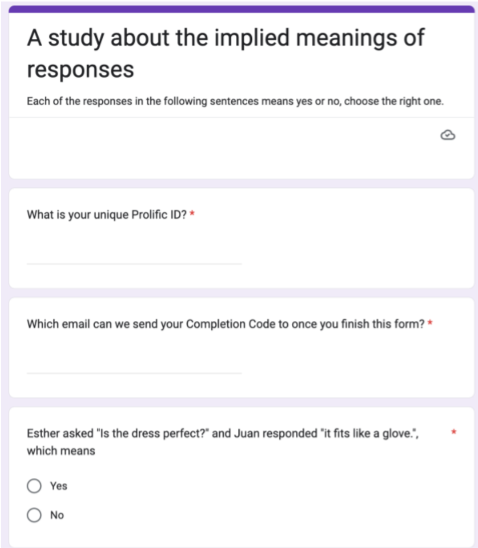
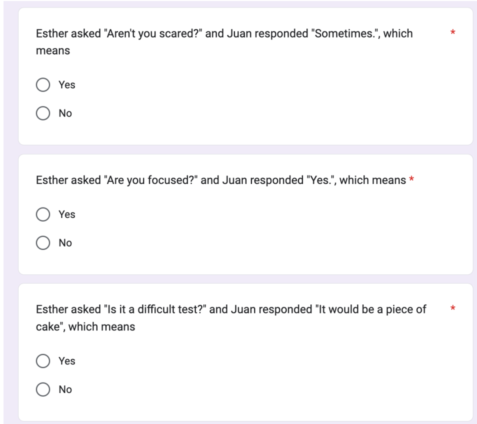


Figure D.1: A screenshot of how the experiment is presented to potential annotators on Prolific (www.prolific.co).

The 150 test examples are wrapped in prompt template 2 (see Table D.1) and presented in a Google form. We opted to wrap all examples in prompt template


A study about the implied meanings of responses
Each of the responses in the following sentences means yes or no, choose the right one.
What is your unique Prolific ID? *
Which email can we send your Completion Code to once you finish this form? *
Esther asked "Is the dress perfect?" and Juan responded "it fits like a glove.", which means *
Yes
No

(a) The start of the Google form participants are asked to fill out for the human study.


Esther asked "Aren't you scared?" and Juan responded "Sometimes.", which means *
Yes
No
Esther asked "Are you focused?" and Juan responded "Yes.", which means *
Yes
No
Esther asked "Is it a difficult test?" and Juan responded "It would be a piece of cake", which means *
Yes
No

(b) Part of the Google form the participants are asked to fill out. The second question in this image is part of the attention test. Juan's response does not contain an implicature but simply gives away the correct answer.

Figure D.2: Screenshots of the Google form participants fill out as part of the implicature study.

2 to make the full human study directly comparable to the model's results on template 2. If we had done a mix of all templates we either had to spent six times as much on the human evalyations (which was not within our budget) or subsample evaluations, making it less comparable to part of the model study. Although models have been shown to be very sensitive to prompt wording, humans are less likely to perform differently for different prompt templates. All templates are coherent natural language that any native English speaker will understand. That said, this is speculative, and to confirm this hypothesis future work should investigate the effect of different wordings on implicature resolution by humans. The participants are asked to choose the correct continuation, yes or no (see Figure D.2a). As recommended by Prolific, we subject the participants to an attention test (see Figure D.2b). At three random places in the form, we add a question that does not contain an implicature and obviously maps to "yes". In this way, if the participants fails at least two of these questions, we can conclude they were not paying attention and remove their answers from the result. In practice, this happened once and we decided to pay the participant regardless, but discard their results, which were close to random.

Table D.6 shows the performance of each annotator on the subset they annotated. The average human performance across subsets and annotators is 86.2% $\pm$ 2.3, the best performance is 89.8% $\pm$ 2.2, and the worst performance is 83.5% $\pm$ 1.5. The column "IAA" shows the average Cohen's Kappa coefficient which is the pairwise inter-annotator agreement for each annotator per subset. All agreements are substantial according to the interpretation guidelines for Cohen's Kappa (between 0.61–0.80).

Table D.6: The performance of the human annotators on the subsets of the test set. Subset 1 through 4 are non-overlapping and cover the whole test set. Annotator X for subset Y might be a different human than annotator X for subset Z. IAA is the average pairwise inter-annotator agreement (Cohen's kappa coefficient) between annotators per subset.

Annotator	1	2	3	4	5	Mean	Best	Worst	IAA
Subset 1	86.0%	92.0%	90.7%	90.6%	86.0%	89.1%	92.0%	86.0%	0.73
Subset 2	84.7%	83.3%	87.3%	86.0%	86.0%	85.5%	87.3%	83.3%	0.64
Subset 3	84.0%	85.3%	88.0%	86.0%	82.7%	85.2%	88.0%	82.7%	0.78
Subset 4	85.3%	82.7%	84.0%	82.0%	92.0%	85.2%	92.0%	82.0%	0.71
Total	-	-	-	-	-	86.2%	89.8%	83.5%	0.72
Std	-	-	-	-	-	2.3	2.2	1.5	0.1

Human source of disagreement with ground-truth. We do an analysis of the source of disagreement with the ground-truth. We explicitly do not call this error, as in some cases examples might allow multiple interpretations, and both could be right. In other cases, the ground-truth might be wrong.

Annotation errors and multiple interpretations: We analyse the examples for which most humans choose a different answer than the ground-truth. For 30 out of 600 examples in the test set, only one or zero people choose the same answer as the ground-truth. Of these examples, most are annotated wrongly (18 of 30). For example: 'Are you busy?', 'I'm drowning in work.', implicature: 'no'. Some are examples that can have multiple different interpretations (12 of 18), and the ground-truth answer likely just chooses one that is unnatural to humans. For example: 'You don't remember them?', 'Leave me alone!', implicature: 'yes'. 6 of the 30 examples are particularised, and 1 is generalised.

Examples for which all humans agree with the ground-truth: There are 409 out of 600 examples that all humans get correct. This set of examples contains most of the generalised implicatures (39 out of 47). These contain 58 out of 94 particularised examples.

Examples most humans agree with the ground-truth: When we look at examples that 3 or more humans got correct, that comprises most of the test set (530 of 600), and all of the generalised examples (47 of 47). This subset has 78 of 94 particularised examples, so for 16 particularised examples 3 or more humans disagree with the ground-truth.

D.5 Comparison with BIG-bench implicatures task

One of the BIG-bench tasks is related to the task in Chapter 5.[1] It uses the same underlying dataset we use [GM20]. With the below we aim to discuss our contribution in light of the BIG-bench result. To summarise; the methodology used by the BIG-bench task contributors has limitations, which call into question the validity of their claims. Further, some of the BIG-bench results are irreproducible due to missing details in the tech report and the use of proprietary models. Considering this, our work is an important contribution validating the BIG-bench results in a reproducible and methodologically sound way, and above that providing insight into what aspects of LLM training are crucial for the ability to do pragmatic inferences.

Limitations of the methodological approach of the task contributors in BIG-bench implicatures. Our benchmark has 30% more data, which the BIG-bench task contributors discard. In this section we motivate the crucial importance of that data for evaluating implicature understanding (Section D.5.1), and why BIG-bench in turn might be overestimating the performance of LLMs on implicature resolution (Section D.5.2). Moreover, the human performance on the BIG-bench task indicates low quality human annotation, which we will also elaborate upon below, noting that this is impossible to verify because the BIG-bench report does not detail how the evaluation was done for this task (Section D.5.3).

D.5.1 Discarding ambiguous examples

The BIG-bench task preprocesses the 1001 examples that [GM20] curate by keeping only yes/no questions, discarding any examples that are ambiguous according to the task contributors, and discarding remaining examples to create a 50/50 distribution of yes/no answers. This leaves them with 492 examples. Of these examples, 81 appear in our development set and the remaining 411 appear in our test set. Our test set has 600 examples, so BIG-bench effectively discarded 189 ambiguous examples compared to our test set; a bit more than 30% of the benchmark. To illustrate the importance of not discarding this data, we cherry picked a few examples that the BIG-bench authors discarded from the data.

- Utterance: “Can you lend me hundred dollars?”, Response: ”Is this supposed to be some kind of a joke?”, Implicature: “No”

[1] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/implicatures

- Utterance: “Do you know, how long is Uncle Arthur staying with us?”, Response: ”Ask your father.”, Implicature: “No”

Indeed, these examples are ambiguous. Asking whether the request for a hundred dollars is a joke does not literally mean you’re saying no to the request. The response “ask your father” does not mean the speaker does not actually know, maybe they just do not want to respond. The humans in our study all infer the intended ground truth implicature. This shows a general property of implicatures; they are ambiguous, but often humans do infer the intended meaning. Ambiguity is not a discrete property. Some examples may be so vague that no one gets it. The following are examples the BIG-bench task discards that the humans in our study did struggle with:

- Utterance: “Got any more of those?”, Response: ”Nothing I’m at liberty to reveal here.”, Implicature: ”Yes”
- Utterance: “Have you finished sight-seeing?”, Response: ”Sorry. I should’ve come to see you first.”, Implicature: “Yes”

In the first of these the implicature is “yes” because the person responding is implying that they do have more, they just cannot reveal them. Otherwise they would most likely simply say no. In the second example it feels more natural that someone says this when they are finished sight-seeing, otherwise they would’ve probably said something to the effect of “I’m still out, but I’m sorry..”. In any case, humans in our study did not understand these responses like that. This illustrates another aspect of implicature; sometimes communication will go wrong over it. Removing implicatures that are ambiguous though, defeats the purpose of the task, as they are all ambiguous to a certain degree. The purpose of this study is to compare how humans resolve this type of non-literal language compared to how models do it. The human baseline of 86% accuracy that humans achieve on our test set deals more naturally with examples that are too ambiguous for models to understand than discarding examples based on the subjective opinion of a few people.

D.5.2 Overestimation of performance on implicature understanding

On the overlapping part of our test set and theirs the humans in our study achieve 92.8% accuracy. The best model on the BIG-bench task is PaLM, achieving a zero-shot performance of 64.4%. Note that this performance is on their full test set (not the overlapping part) and hence not directly comparable. Nonetheless, the missing examples are randomly sampled for our development set, and we can be pretty confident this number indicates a large gap with human performance.

Two-shot PaLM comes very close to human performance with 91.7% accuracy, but of course this does not take into account the 189 more challenging examples that are part of our benchmark. Humans achieve 71.9% performance on this subset of ambiguous data, indicating that these data are more difficult than average, but nonetheless performance is higher than random. Without access to the models used to evaluate the BIG-bench task we cannot say anything for certain, but we expect the performance of PaLM to be overestimated as it does not account for a large part of the type of implicatures found in natural discourse; ambiguous ones.

D.5.3 Other limitations

Poor quality human evaluation. The average human evaluator on BIG-bench implicatures achieves around 82% performance (where ours achieves on average 86% on a more challenging dataset), and their human best rater achieves 100% (where our human best is 92%). This difference between human average and best hints at poor quality average rating. This is impossible to verify because there is no information in the BIG-bench tech report on how the human evaluation was done exactly, or even which examples where evaluated.

No fine-tuned models. BIG-bench uses only base LLMs and no SotA fine-tuning methods. A question that remains is therefore, what aspects of LLMs contribute to their performance on implicatures? In our work we find that implicature performance emerges at a much smaller scale in models instruction fine-tuned at the example level, and that scale and prompting techniques are important.

D.6 Chain-of-thought completions by GPT-4

GPT-4 reaches human-level performance with CoT prompting. In this section we look at a few completions to help illuminate how CoT-prompting improves implicature resolution.

Specifically, we look at some of the examples for which GPT-4 5-shot gets 0% accuracy (i.e. wrong for all templates), and GPT-4 5-shot CoT gets 100% accuracy (i.e. right for all templates). This is a set of 10 examples. We only look at the first template, but all templates get a similar performance with CoT.

Utterance: 'Do you sell calculators?'
Response: 'We are all sold out. We wont probably have any more in for a few weeks.'
Implicature: Yes. But we dont have any calculators to sell now.

This is an example that GPT-4 gets wrong for all templates 5-shot, probably because the response has a negative sentiment, and also it means they do not sell calculators right now. However, the implicature is yes, because they do sell them normally. The CoT completion is:

CoT completion: 'Alice says they are all sold out and won't have any more in for a few weeks. Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. Alice must be implying that they do sell calculators, but they currently don't have any in stock. The answer to Bob's question is yes. Answer: yes'

This example is similar to the one given in Section 5.3, and there are more like these in the dataset: the sentiment is negative, and an interpretation can be negative, but strictly the implicature is positive (yes).

Utterance: 'Do you know where he is?'
Response: 'I'll find him.'
Implicature: No.
GPT-4 gets this wrong for all templates 5-shot, and right for all of them with CoT:

CoT completion: 'Alice says she will find him. Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. If Alice knew where he is, she would have provided that information directly. Instead, she says she will find him, implying that she does not currently know his location. The answer to Bob's question 'Do you know where he is?' must therefore be no. Answer: no'

It's unclear why GPT-4 gets this one wrong 5-shot (all humans get this one right), but the CoT is very relevant and helpful.

Utterance: 'It seems Cathy is not happy at all with her new room mate. Did you ask her why?'
Response: 'Believe me. I tried to find out. But I simply could not pin it down.'
Implicature: Yes. But Cathy did not give specific reasons for her feelings.

Again an example where the sentiment of the response seems to hint at no. Of 5 human annotators, also only 2 interpret this response as "yes". With a CoT, GPT-4 gets it right six times:

CoT completion: 'Alice says she tried to find out but couldn't pin it down. Alice's response must be relevant to Bob's question because Alice is a cooperative conversational partner. Alice must mean that she attempted to ask Cathy why she is unhappy with her new roommate, but was unable to get a clear answer. The answer to Bob's question 'Did you ask her why?' must therefore be yes. Answer: yes'

A helpful reasoning trace. All CoT completions by the models we have run CoT on are available in the GitHub: `https://github.com/LauraRuis/do-pigs-fly`.

D.7 Additional results

D.7.1 Contrastive experiment

In this section we reframe the implicature resolution task to a contrastive one, allowing the model to contrast the coherent to the incoherent sentence in a single prompt.

Contrastive task. In the ranking task the model is required to assign higher likelihood to the coherent utterance than the incoherent one ($p_\theta(y) > p_\theta(\hat{y})$). In assigning a likelihood to y, the model has no knowledge of $\hat{y}$, and vice-versa. We hypothesize that the task might become easier if we reformulate it as a contrastive task. Consider the following prompt y_p.

> Which of the following sentences is coherent:
>
> A: Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means no.
>
> B: Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means yes.
>
> Answer:

We can now evaluate the models' ability to understand which is the coherent sentence by evaluating whether it assigns $p_\theta(A \mid y_p) > p_\theta(B \mid y_p)$. Note that this can again be framed in a ranking task of assigning a higher likelihood to the coherent prompt. If we finish the above prompt y_p by adding "A" to make a coherent prompt y and "B" to make an incoherent prompt $\hat{y}$ we can again formulate the task by $p_\theta(y) > p_\theta(\hat{y})$. The difference is that within both the coherent and the incoherent prompt, the model can contrast the coherent and incoherent utterance to each other. We randomise the assignment of A and B to the utterances.

We do a small experiment with the contrastive task with one of the best performing models overall, OpenAI's text-davinci-002, for $k = \{0, 1, 5\}$. We use two prompt templates and for each template try three different multiple choice answers: A and B like above, one and two, or the full text of the answer. For the last option the coherent prompt $\mathbf{x}$ would look as follows:

Which of the following sentences is coherent:

A: Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means no.

B: Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means yes.

Answer: Esther asked "Can you come to my party on Friday?" and Juan responded "I have to work", which means no.

Table D.7: Performance on the implicature task framed contrastively by OpenAI's text-davinci-002. The mean and standard deviation are reported over two different prompt templates (template 1 and 2).

k	Non-contrastive	Rank one, two	Rank A, B	Rank full text
0	71.3% ± 1.75	53.9% ± 0.9	59.3% ± 1.3	48.9% ± 0.6
1	76.1% ± 2.6	59.4% ± 1.6	63.2% ± 2.0	66.9% ± 0.9
5	80.5% ± 2.3	61.4% ± 1.3	64.0% ± 1.3	67.9% ± 2.1

In Table D.7, perhaps surprisingly, we can see that the contrastive task is much more difficult than the original ranking task. For $k = 0$, the result is random except for the prompt where the multiple choice options are A and B. For $k = \{1, 5\}$ the full text ranking does best, but is still significantly worse than the original ranking setup. Because of these disappointing results, we did not evaluate the other models contrastively. Future work must establish whether the contrastive setup is worse across all model classes and sizes.

D.7.2 Variance over prompt ordering

As mentioned in Section 5.2, models are sensitive to the ordering of the k examples in the prompt. Instead of marginalising over this random factor by evaluating all possible prompt orderings, we randomly sampled an ordered set of examples from the development set for each test example. Throughout experiments, we kept this randomly sampled order the same, meaning if you re-run the 5-shot evaluation you get exactly the same orderings. The reason for this is that we want evaluate each model equally. In this section we ask how the performance chances for the best performing model if we select another random order. We do this for the 5-shot evaluation, because the results show that adding more in-context examples barely helps performance.

Table D.8: Variance over prompt ordering for 5-shot evaluation per prompt template (P.T.) for text-davinci-002

Seed	P. T. 1	P. T. 2	P. T. 3	P. T. 4	P. T. 5	P. T. 6	Mean
0	80.17	78.17	82.83	80.50	79.17	76.50	79.56
1	80.17	76.17	81.33	81.83	76.00	76.33	78.64
2	79.50	78.17	81.17	80.17	78.17	76.50	78.94
mean	79.94	77.50	81.78	80.83	77.78	76.44	-
std	0.31	0.94	0.75	0.72	1.32	0.08	-

Table D.8 shows the results of this experiment. Some prompt templates seem to be more sensitive to prompt example ordering than others, but for none of them the variance is high enough to change any conclusions.

D.7.3 Different zero-shot instruction prompts

There is a narrative around large language models that if they fail a task, it might be that the prompt was not the right one (through works like Reynolds and McDonell [RM21b] and Kojima et al. [Koj+22]). The idea is that they can be prompted to simulate almost anything, if you set them up correctly. Because implicature resolution is a ubiquitous result of learning language, we hold the view that a model should be able to do this task if a prompt is given in coherent natural language. Nonetheless, in an additional effort to find the "let's think step-by-step" [Koj+22] of zero-shot implicature resolution we try three more prompt templates.

We evaluate a base large language model and two instructable models: GPT-3-175B, text-davinci-001, and text-davinci-002. The prompts we use are taken from recent work that proposes a dialogue agent trained with human feedback [Gla+22], but adapted to the task of implicature resolution. The full prompts are presented in Table D.3 and Table D.9 shows the results. The new templates do not improve performance for any of these models. The variance over the prompt templates for text-davinci-002 is high, and the best prompt template of these three does achieve a slightly higher accuracy than the others: 74.5%. These results do not change the picture sketched so far.

Table D.9: Zero-shot accuracy over three additional prompt templates for a base LLM and two instructable models.

Model	Templates
GPT-3-175b	59.2% ± 4.5
text-davinci-001-⋆	66.1% ± 3.2
text-davinci-002-⋆	67.7% ± 9.6

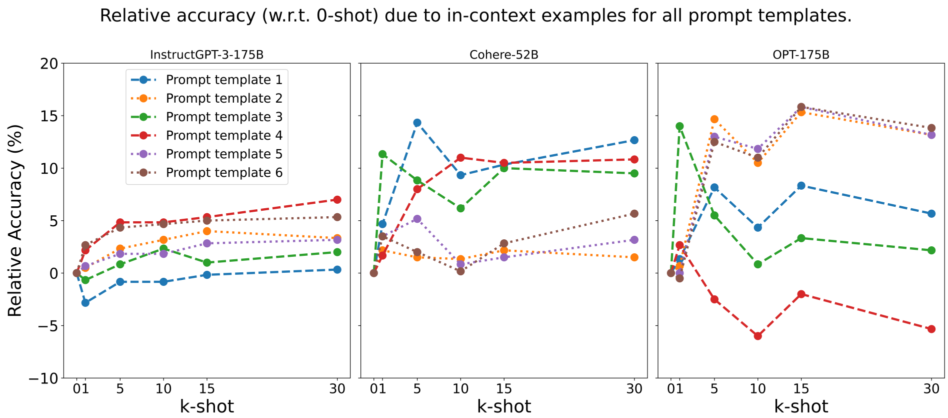


Figure D.3: Relative performance increase over 0-shot due to in-context prompting. Structured prompt templates are dashed lines (1, 3, 4) and natural prompt templates dotted lines (2, 5, 6).

D.7.4 The effect of in-context examples on sensitivity to prompt wording

Figure D.3 shows the relative performance increase due to in-context prompting broken down per prompt template. For text-davinci-001, most templates benefit similarly from more in-context examples, except for template 1. Perhaps surprisingly, we see that this template already achieves a performance of 76.5% at the zero-shot evaluation and does not improve much with few-shot prompting. For Cohere-52B and OPT-175B we see a clear grouping between the structured prompts (dashed lines) and natural prompts (dotted lines). Cohere struggles significantly more with the structured prompts than with the natural prompts in the zero-shot evaluation, and few-shot prompting can mitigate that, lowering the standard deviation over prompt templates to 1.89 at $k = 30$ from 4 at $k = 0$. OPT benefits from prompting for the natural prompts, but not for the structured prompts.

D.7.5 Variance over API runs

In this section we comment on the reproducibility of research done using APIs. OpenAI and Cohere have their models behind an API, meaning we do not have control over what happens to the prompt before the model processes it. We run the zero-shot evaluation ten more times for two models of OpenAI and Cohere, text-davinci-002 and Cohere-52B. The results from this experiment are shown in Table D.10 and D.11. From this we can conclude that there is some stochasticity in the API that we have no control over, a bit more for OpenAI than for Cohere, but again we can be relatively confident that the conclusion will not be different because of it. The results from this work are therefore reproducible with access to the same models behind the API now. Unfortunately, when OpenAI or Cohere changes the models behind the API, these results are not exactly reproducible anymore.

For completeness, we add the timestamp that each result was obtained below (Appendix D.8).

Table D.10: Results per prompt template (P.T.) for 10 different runs from text-davinci-002 for 0-shot evaluation.
Each evaluation has exactly the same text, so the variance in performance is due to API stochasticity.

API-run	**P. T. 1**	**P. T. 2**	**P. T. 3**	**P. T. 4**	**P. T. 5**	**P. T. 6**	**Mean**
0	73.50	68.83	73.00	71.17	67.17	68.83	70.42
1	73.83	69.00	72.83	71.50	67.67	68.33	70.53
2	73.67	68.67	73.17	71.33	67.50	68.50	70.47
3	73.83	68.17	73.17	71.00	67.67	68.17	70.33
4	73.67	68.83	73.33	71.17	67.00	68.33	70.39
5	73.83	68.50	73.00	71.00	67.00	68.17	70.25
6	73.67	69.00	73.00	71.17	67.33	68.50	70.44
7	73.67	68.67	72.83	71.33	67.50	68.67	70.44
8	73.83	69.17	72.83	71.17	67.33	68.00	70.39
9	73.50	68.50	72.83	71.00	67.50	68.67	70.33
10	73.67	69.50	73.00	71.33	67.50	68.50	70.58
mean	73.70	68.80	73.00	71.20	67.38	68.42	-
std	0.12	0.35	0.16	0.16	0.23	0.24	-

Table D.11: Results per prompt template (P.T.) for 10 different runs from Cohere-52B for 0-shot evaluation.
Each evaluation has exactly the same text, so the variance in performance is due to API stochasticity.

API-run	**P. T. 1**	**P. T. 2**	**P. T. 3**	**P. T. 4**	**P. T. 5**	**P. T. 6**	**Mean**
0	56.00	62.67	54.33	54.00	62.17	62.17	58.56
1	56.00	62.83	54.33	54.00	62.33	62.33	58.64
2	56.00	62.83	54.33	54.00	62.17	62.33	58.61
3	56.00	62.83	54.33	54.00	62.17	62.33	58.61
4	55.83	62.67	54.33	54.00	62.17	62.33	58.56
5	56.00	62.83	54.33	54.00	62.17	62.17	58.58
6	56.00	62.83	54.33	54.00	62.17	62.17	58.58
7	56.00	62.67	54.33	54.00	62.33	62.17	58.58
8	56.00	62.83	54.33	54.00	62.00	62.33	58.58
9	56.00	62.83	54.00	53.83	62.17	62.17	58.50
mean	55.98	62.78	54.30	53.98	62.18	62.25	-
std	0.05	0.08	0.10	0.05	0.09	0.08	-

D.7.6 Experiment with random in-context labels

Chapter 5 presents the thesis that instruction-tuning at the example level ("Example IT") is important for pragmatic understanding in LLMs. However, the 0-shot result that one of the models in the Example IT group achieves is similar to that of base models; Cohere-command-52b obtains a zero-shot performance of 60.2%. From the sharp rise in performance observed for the $k = 0$ to $k = 1$ result (from 60.2% to 72.8%) we hypothesise that the k-shot in-context examples in this task do not necessarily teach the model pragmatics in-context, but prime the model for the task format (namely, outputting either "yes" or "no" as detailed in Section 5.2). If this hypothesis is true, we would observe similar performance regardless of whether the labels given in the prompt for the few-shot examples are true. We test this empirically for two base models (GPT-3, Cohere-52b) and two Example IT models (text-davinci-001, Cohere-command-52b) for 1-shot and 5-shot evaluation. The results can be found in Table D.12. We find that for the Example IT models in-context prompts with random labels obtain the same results (i.e. within confidence intervals) as the experiments with ground-truth labels in the in-context examples. For base models however we do observe a drop in performance; for GPT-3-175b at 5-shot, and Cohere-52b both at 1- and 5-shot. Taken together, we can conclude that for base models the content of the in-context prompt seems important, whereas for models in the example IT group the in-context examples mainly serve as a primer for the task structure.

Table D.12: The results of the 1- and 5-shot experiment with random labels for the few-shot examples as opposed to the the true labels. We find that performance does not degrade for the models in the Example IT group, which implies that for these models not the content of the examples is important for performance, but the structure.

Model	1-shot	1-shot rand labels	5-shot	5-shot rand labels
GPT-3-175b	65.7% $\pm$ 1.4	65.4% $\pm$ 1.2	68.7% $\pm$ 1.5	64.7% $\pm$ 1.9
Cohere-52b	63.0% $\pm$ 3.8	58.3% $\pm$ 3.3	65.1% $\pm$ 2.9	60.5% $\pm$ 1.9
text-davinci-001	72.7% $\pm$ 1.3	73.9% $\pm$ 1.7	74.5% $\pm$ 1.0	73.4% $\pm$ 1.2
Cohere-command-52b	72.8% $\pm$ 1.3	72.0% $\pm$ 1.6	75.4% $\pm$ 1.8	73.5% $\pm$ 2.7

Table D.13: Results of the chain-of-thought (CoT) experiment for models in the base group. The numbers between brackets show the difference in performance with the number on the same row one column to the left. These models do not benefit from CoT-prompting. The reason Cohere-6b achieves such a low score for CoT-prompting is because it is not able to adhere to the correct output format (yes/no).

Model	0-shot	5-shot	5-shot CoT
GPT-3-350m	51.5% ± 3.0	55.7% ± 1.6 (+4.2%)	55.0% ± 3.5 (-0.7%)
GPT-3-1.3b	57.7% ± 3.1	62.6% ± 2.0 (+4.9%)	54.4% ± 5.8 (-8.2%)
GPT-3-6.7b	54.8% ± 1.9	62.4% ± 1.5 (+7.6%)	61.0% ± 2.3 (+4.0%)
GPT-3-175b	57.2% ± 4.4	68.7% ± 1.5 (+11.5%)	60.3% ± 4.2 (-8.4%)
Cohere-6b	57.3% ± 2.2	60.9% ± 4.1 (+3.6%)	29.2% ± 14.7 (-31.7%)
Cohere-52b	58.5% ± 4.0	65.1% ± 2.9 (+6.6%)	64.7% ± 3.2 (-0.4%)

D.7.7 Chain-of-thought on base models

In Section 5.3 we do a CoT experiment on the models in the Example IT group. Base models also benefit from in-context examples, so it makes sense to also try CoT prompting on these models. After attempting this for two of the model classes in the group, we decided not to apply this prompting technique to the other models, because it decreases performance, sometimes significantly. See the results of the CoT experiment on the two base model classes in Table D.13.

D.7.8 Testing for spurious correlations

In this section, we do a small scale experiment to test whether the benchmark has spurious correlations. Specifically, we run the benchmark with only the utterance or only the response as input. Strictly, getting the implicature right from the response only does not always indicate spurious correlations, as some examples only need the response (e.g. rhetorical questions like 'do pigs fly?'). Utterance-only results do always indicate spurious correlations. We run this experiment for GPT-3.5-turbo and GPT-4 0-shot and 5-shot (see Table D.14 and Table D.15).

Table D.14: Results of running the benchmark with only the utterance as input, to test for spurious correlations with the label.

Utterance-only	0-shot	5-shot
GPT-3.5-Turbo	54.3% ± 3.3	41.7% ± 12.4
GPT-4	48.9% ± 10.5	53.7% ± 0.5

Table D.15: Results of running the benchmark with only the response as input, to test what part of the examples can be resolved without the utterance.

Response-only	**0-shot**	**5-shot**
GPT-3.5-Turbo	59.2% ± 4.7	58.3% ± 6.6
GPT-4	62.6% ± 1.7	65.5% ± 1.1

We find that models mostly perform random for utterance-only, so spurious correlations do not seem to be an issue. For response-only, GPT-4 5-shot gets 65% accuracy. Some examples it gets right are: "do fish swim?" and "let's hope so".

Table D.16: An example from the dataset for each type of implicature found in the test set. The rightmost column shows the amount of that type we manually found in the test set.

Type	Example Utterance	Example Response	Impl.	#
Generalised	You know all these people?	Some.	No.	47
Particularised	Want to stay for a nightcap?	I've gotta get up early.	No.	94
World knowledge	Did you leave fingerprints?	I wore gloves.	No.	23
Idiom	Would he fire me?	He's all bark and no bite.	No.	42
Rhetorical question	Can you drive that far?	Can fish swim?	Yes.	11
Other	-	-	-	383

D.7.9 Detailed results type label analysis

In Section 5.3 we do an analysis of two types of examples that occur frequently in the dataset, namely generalised and particularised implicatures. Here, we detail the full taxonomy of types of examples occurring in the dataset and report detailed results for each type per model (see D.17 until Table D.30 below). In Table D.16 the full taxonomy of the examples is shown, representing types of examples that occur frequently in the dataset. We manually labeled 217 examples of the 600 examples in the test set according to this taxonomy. The remaining 383 examples do not fall as clearly within a category and are grouped together as type *other*. *Generalised* implicatures require little or no context to be understood. They are the simplest type of example in the test set, and generally imply the same thing ("some" almost always implies "not all"). *Particularised* implicatures, by contrast, do require context to be resolved. For example, from Table D.16, we need the context that it is undesirable to stay up late drinking when one has to get up early (see in Appendix D more on generalised vs. particularised). The type *world knowledge* requires knowledge of the physical world to be resolved. From the example in Table D.16; we need to know that you cannot leave fingerprints when wearing gloves to resolve this implicature. *Idiom* types contain an idiom or a metaphor that one needs to know or understand to resolve the implicature, and finally *Rhetorical question* types contain a question like "Is the Pope Catholic?", often requiring factual knowledge to be resolved.

The following tables contain the detailed results broken down per example type: Table D.17 - Table D.30. The most interesting pattern in this data is that for almost all models, even the best model (GPT-4 30-shot in Table D.28), there is a significant gap between human-level performance on the particularised examples. This gap is larger than the gap for the other labels usually. Few-shot prompting can often mitigate this (e.g. for GPT-3-175b, Cohere-52b, and text-davinci-002), but not always (e.g. for GPT-4 the gap remains large for $k = 30$). However, for GPT-4, chain-of-thought can mitigate the gap as seen in Table D.30. Where GPT-4 30-shot obtains 71.97% accuracy on the particularised examples (and humans 83.18%), GPT-4 with 5-shot

CoT achieves 81.63%, which is close to human-level. We find that the particularised examples mostly benefit from CoT prompting. Namely, for the generalised type of examples, GPT-4 30-shot already achieves 86.23% accuracy and CoT improves this to 88.66%, which is a much smaller improvement than for the particularised examples.

Table D.17: Accuracy per label for 0-shot evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
OPT-125m	50.92	50.00 +/- 2.17	51.52 +/- 9.96	57.58 +/- 10.05
OPT-350m	57.14	57.97 +/- 10.25	64.77 +/- 3.65	65.15 +/- 3.39
OPT-1.3b	60.36	60.14 +/- 5.84	68.94 +/- 5.52	59.09 +/- 4.55
OPT-2.7b	59.56	60.87 +/- 6.15	67.05 +/- 2.18	69.70 +/- 6.78
OPT-6.7b	60.33	59.42 +/- 6.95	59.47 +/- 2.04	53.03 +/- 19.93
OPT-13b	61.03	63.77 +/- 14.78	73.86 +/- 7.51	66.67 +/- 16.32
OPT-30b	61.47	65.94 +/- 10.48	62.88 +/- 8.05	74.24 +/- 6.25
OPT-66b	61.33	69.57 +/- 13.75	60.23 +/- 4.30	59.09 +/- 18.74
OPT-175b	55.33	55.07 +/- 5.42	54.55 +/- 9.19	63.64 +/- 21.64
BLOOM-560m	51.58	54.35 +/- 5.47	54.92 +/- 16.72	50.00 +/- 13.64
BLOOM-1b1	51.17	50.00 +/- 2.17	50.38 +/- 11.77	53.03 +/- 12.22
BLOOM-1b7	53.61	52.17 +/- 6.15	53.79 +/- 8.77	68.18 +/- 6.94
BLOOM-3b	56.89	54.35 +/- 6.02	59.85 +/- 4.48	63.64 +/- 5.25
BLOOM-7b1	58.67	63.77 +/- 14.57	68.94 +/- 5.82	68.18 +/- 4.55
BLOOM-176b	54.22	55.07 +/- 7.39	50.38 +/- 11.01	62.12 +/- 9.70
EleutherAI-125m	51.89	56.52 +/- 9.72	52.65 +/- 8.84	63.64 +/- 5.25
EleutherAI-1.3b	53.14	51.45 +/- 3.90	53.03 +/- 11.19	62.12 +/- 3.39
EleutherAI-2.7b	59.17	60.14 +/- 13.38	65.91 +/- 3.94	68.18 +/- 4.55
EleutherAI-6b	56.36	57.25 +/- 7.28	56.06 +/- 8.87	50.00 +/- 17.99
EleutherAI-20b	57.53	51.45 +/- 3.90	67.80 +/- 5.93	72.73 +/- 5.25
Cohere-409m	51.61	52.17 +/- 4.35	53.41 +/- 11.94	54.55 +/- 12.86
Cohere-6b	57.28	55.80 +/- 5.28	60.23 +/- 5.98	72.73 +/- 9.09
Cohere-13b	57.19	59.42 +/- 4.81	54.55 +/- 10.82	48.48 +/- 10.05
Cohere-52b	58.50	60.14 +/- 13.61	65.15 +/- 3.86	74.24 +/- 11.03
GPT-3-350m	51.47	51.45 +/- 3.90	53.41 +/- 13.56	50.00 +/- 13.64
GPT-3-1.3b	57.72	61.59 +/- 11.06	64.39 +/- 4.08	65.15 +/- 3.39
GPT-3-6.7b	54.83	54.35 +/- 6.99	53.79 +/- 7.61	62.12 +/- 3.39
GPT-3-175b	57.22	55.80 +/- 7.28	68.94 +/- 5.19	77.27 +/- 8.70
T0-3b	48.25	54.35 +/- 4.86	42.42 +/- 4.29	36.36 +/- 0.00
T0-11b	55.61	60.14 +/- 6.84	54.92 +/- 14.93	36.36 +/- 0.00
BlenderBot-90m	46.64	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-3b	53.44	47.83 +/- 0.00	61.36 +/- 1.31	63.64 +/- 0.00
BlenderBot-9b	53.36	52.17 +/- 6.64	60.98 +/- 4.81	63.64 +/- 0.00
Flan-T5-780m	63.31	72.46 +/- 4.10	71.97 +/- 5.82	54.55 +/- 13.89
Flan-T5-3b	52.50	50.72 +/- 6.95	51.89 +/- 4.23	42.42 +/- 8.57
Flan-T5-11b	60.78	65.94 +/- 5.84	72.35 +/- 7.59	65.15 +/- 6.25
Cohere-command-6b	66.31	72.46 +/- 7.80	78.41 +/- 4.30	37.88 +/- 3.39
Cohere-command-52b	60.22	66.67 +/- 10.85	63.64 +/- 10.33	77.27 +/- 6.94
text-ada-001-unknown	56.50	63.77 +/- 4.10	58.71 +/- 16.04	51.52 +/- 10.05
text-babbage-001-unknown	64.47	67.39 +/- 6.02	76.52 +/- 1.69	60.61 +/- 10.05
text-curie-001-unknown	68.94	76.81 +/- 3.24	76.89 +/- 2.76	54.55 +/- 12.86
text-davinci-001-unknown	72.31	84.78 +/- 7.43	78.79 +/- 4.08	59.09 +/- 13.64
text-davinci-002-unknown	70.58	82.61 +/- 9.05	75.38 +/- 3.05	57.58 +/- 16.32
text-davinci-003-unknown	71.25	86.96 +/- 13.28	72.35 +/- 7.35	48.48 +/- 8.57
ChatGPT-unknown	72.08	82.61 +/- 12.04	83.33 +/- 5.97	56.06 +/- 16.11
GPT-4-unknown	81.78	92.03 +/- 2.99	90.91 +/- 3.21	84.85 +/- 8.57
Humans	86.23	93.04	92.73	92.73

D.7.10 Detailed results per model

This section contains the results used for the zero-shot and few-shot evaluation in Section 5.3, broken down per prompt template. See Table D.31 until Table D.80.

Table D.18: Accuracy per label for 0-shot evaluation.

Model	Mean	Particularised	Generalised	Other
OPT-125m	50.92	49.43 +/- 5.52	55.07 +/- 21.10	50.56 +/- 1.33
OPT-350m	57.14	47.92 +/- 4.37	69.20 +/- 8.36	56.68 +/- 4.99
OPT-1.3b	60.36	51.52 +/- 6.81	74.64 +/- 2.05	59.65 +/- 3.51
OPT-2.7b	59.56	50.19 +/- 5.06	69.93 +/- 5.53	59.22 +/- 6.14
OPT-6.7b	60.33	52.27 +/- 5.90	75.36 +/- 2.71	60.77 +/- 6.13
OPT-13b	61.03	55.49 +/- 8.79	75.00 +/- 5.72	58.79 +/- 5.51
OPT-30b	61.47	54.55 +/- 4.15	71.38 +/- 5.94	61.11 +/- 2.12
OPT-66b	61.33	55.11 +/- 7.33	69.93 +/- 12.26	61.46 +/- 3.68
OPT-175b	55.33	54.17 +/- 7.70	58.33 +/- 18.51	55.12 +/- 4.21
BLOOM-560m	51.58	50.76 +/- 5.59	48.91 +/- 25.22	51.59 +/- 3.50
BLOOM-1b1	51.17	50.57 +/- 6.32	53.26 +/- 27.40	51.16 +/- 2.41
BLOOM-1b7	53.61	50.38 +/- 7.95	59.78 +/- 18.82	53.23 +/- 1.62
BLOOM-3b	56.89	51.70 +/- 8.27	67.39 +/- 10.35	56.46 +/- 4.39
BLOOM-7b1	58.67	46.59 +/- 2.86	79.35 +/- 2.74	57.11 +/- 4.03
BLOOM-176b	54.22	54.73 +/- 10.61	60.14 +/- 16.73	53.57 +/- 1.65
EleutherAI-125m	51.89	50.38 +/- 5.71	57.25 +/- 20.15	50.90 +/- 0.99
EleutherAI-1.3b	53.14	50.57 +/- 7.03	55.43 +/- 23.20	53.32 +/- 2.05
EleutherAI-2.7b	59.17	50.95 +/- 7.43	74.64 +/- 4.97	58.10 +/- 2.92
EleutherAI-6b	56.36	53.22 +/- 6.86	69.20 +/- 7.36	55.73 +/- 2.43
EleutherAI-20b	57.53	49.43 +/- 6.68	72.83 +/- 5.99	56.20 +/- 3.01
Cohere-409m	51.61	51.33 +/- 4.84	52.54 +/- 22.71	51.25 +/- 2.93
Cohere-6b	57.28	51.52 +/- 6.68	64.49 +/- 16.06	57.06 +/- 3.13
Cohere-13b	57.19	52.27 +/- 5.83	68.12 +/- 10.99	57.45 +/- 3.35
Cohere-52b	58.50	51.52 +/- 7.21	73.91 +/- 5.75	56.85 +/- 3.81
GPT-3-350m	51.47	50.76 +/- 6.96	52.90 +/- 24.07	51.29 +/- 1.63
GPT-3-1.3b	57.72	50.00 +/- 6.29	67.75 +/- 9.84	57.06 +/- 3.78
GPT-3-6.7b	54.83	52.65 +/- 8.16	63.41 +/- 15.14	54.26 +/- 2.12
GPT-3-175b	57.22	53.03 +/- 1.93	71.01 +/- 4.81	54.61 +/- 5.58
T0-3b	48.25	55.68 +/- 0.66	27.17 +/- 2.74	49.83 +/- 1.90
T0-11b	55.61	57.95 +/- 2.18	47.10 +/- 17.15	56.33 +/- 6.49
BlenderBot-90m	46.64	55.49 +/- 0.42	23.91 +/- 0.00	48.32 +/- 0.00
BlenderBot-3b	53.44	44.51 +/- 0.42	76.09 +/- 0.00	51.81 +/- 0.20
BlenderBot-9b	53.36	49.24 +/- 4.81	71.01 +/- 5.71	51.03 +/- 1.63
Flan-T5-780m	63.31	59.28 +/- 3.90	68.84 +/- 7.60	62.23 +/- 3.13
Flan-T5-3b	52.50	54.55 +/- 1.61	48.19 +/- 11.73	53.14 +/- 3.15
Flan-T5-11b	60.78	51.52 +/- 3.39	73.19 +/- 7.28	59.60 +/- 2.18
Cohere-command-6b	66.31	58.90 +/- 3.62	73.19 +/- 2.71	66.15 +/- 2.41
Cohere-command-52b	60.22	55.49 +/- 4.66	60.51 +/- 16.67	59.99 +/- 5.09
text-ada-001-unknown	56.50	52.65 +/- 3.86	61.59 +/- 15.96	56.24 +/- 5.74
text-babbage-001-unknown	64.47	56.25 +/- 2.52	72.46 +/- 9.86	63.87 +/- 1.55
text-curie-001-unknown	68.94	66.48 +/- 2.34	68.84 +/- 5.98	68.48 +/- 3.63
text-davinci-001-unknown	72.31	59.66 +/- 5.07	79.35 +/- 9.78	73.17 +/- 2.54
text-davinci-002-unknown	70.58	64.20 +/- 3.75	80.07 +/- 5.67	69.94 +/- 3.69
text-davinci-003-unknown	71.25	63.64 +/- 1.86	82.25 +/- 4.77	71.23 +/- 2.74
ChatGPT-unknown	72.08	68.75 +/- 2.99	69.57 +/- 11.16	71.66 +/- 5.79
GPT-4-unknown	81.78	71.59 +/- 3.47	89.86 +/- 2.05	81.35 +/- 1.66
Humans	86.23	83.18	92.17	84.86

Table D.19: Accuracy per label for 1-shot evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
OPT-125m	52.72	43.48 +/- 5.02	54.92 +/- 7.24	59.09 +/- 4.55
OPT-350m	52.92	39.86 +/- 1.62	48.11 +/- 2.76	59.09 +/- 6.94
OPT-1.3b	56.31	54.35 +/- 4.16	58.33 +/- 4.48	53.03 +/- 12.22
OPT-2.7b	56.83	64.49 +/- 15.35	64.39 +/- 2.51	66.67 +/- 4.29
OPT-6.7b	60.08	61.59 +/- 13.84	68.94 +/- 6.90	56.06 +/- 6.25
OPT-13b	60.56	68.84 +/- 7.70	69.70 +/- 6.11	54.55 +/- 15.75
OPT-30b	60.33	71.74 +/- 5.47	63.26 +/- 4.81	51.52 +/- 10.05
OPT-66b	63.19	70.29 +/- 11.06	62.50 +/- 2.86	48.48 +/- 13.55
OPT-175b	58.36	63.77 +/- 4.81	66.67 +/- 8.77	57.58 +/- 17.14
BLOOM-560m	54.83	50.00 +/- 5.47	64.02 +/- 4.62	63.64 +/- 0.00
BLOOM-1b1	52.56	56.52 +/- 9.05	59.47 +/- 2.04	59.09 +/- 4.55
BLOOM-1b7	52.81	54.35 +/- 6.52	60.98 +/- 3.57	63.64 +/- 5.25
BLOOM-3b	55.94	50.72 +/- 4.10	64.39 +/- 4.08	59.09 +/- 4.55
BLOOM-7b1	57.00	50.00 +/- 3.32	64.77 +/- 2.86	62.12 +/- 6.25
BLOOM-176b	61.11	77.54 +/- 3.90	66.67 +/- 6.11	50.00 +/- 6.94
EleutherAI-125m	51.67	44.93 +/- 6.95	50.76 +/- 4.29	57.58 +/- 4.29
EleutherAI-1.3b	55.72	47.10 +/- 4.64	55.68 +/- 4.50	50.00 +/- 11.44
EleutherAI-2.7b	55.50	54.35 +/- 5.47	67.42 +/- 4.67	65.15 +/- 9.70
EleutherAI-6b	54.97	57.25 +/- 5.84	60.23 +/- 4.30	53.03 +/- 8.16
EleutherAI-20b	55.86	69.57 +/- 4.35	62.88 +/- 4.85	53.03 +/- 6.25
Cohere-409m	51.89	42.75 +/- 6.84	51.89 +/- 4.62	54.55 +/- 5.25
Cohere-6b	57.86	58.70 +/- 12.74	67.05 +/- 5.83	68.18 +/- 11.44
Cohere-13b	61.78	71.74 +/- 11.43	67.42 +/- 8.47	37.88 +/- 9.70
Cohere-52b	62.97	66.67 +/- 6.48	70.08 +/- 4.43	62.12 +/- 14.29
GPT-3-350m	55.97	50.72 +/- 4.10	61.74 +/- 5.15	69.70 +/- 12.49
GPT-3-1.3b	60.75	58.70 +/- 4.16	65.53 +/- 4.23	54.55 +/- 5.25
GPT-3-6.7b	61.17	60.87 +/- 11.77	69.32 +/- 3.65	56.06 +/- 8.16
GPT-3-175b	65.72	76.81 +/- 3.24	73.48 +/- 2.51	57.58 +/- 16.32
T0-3b	48.89	54.35 +/- 2.17	42.80 +/- 2.04	36.36 +/- 0.00
T0-11b	47.78	52.17 +/- 0.00	40.53 +/- 2.43	36.36 +/- 0.00
BlenderBot-90m	49.94	55.07 +/- 6.48	47.73 +/- 9.99	51.52 +/- 13.55
BlenderBot-3b	53.31	47.83 +/- 0.00	61.36 +/- 0.00	63.64 +/- 0.00
BlenderBot-9b	52.53	50.72 +/- 9.61	57.20 +/- 12.95	66.67 +/- 6.78
Flan-T5-780m	62.89	64.49 +/- 7.70	67.42 +/- 13.03	46.97 +/- 8.16
Flan-T5-3b	52.75	65.22 +/- 15.47	55.30 +/- 9.34	45.45 +/- 12.86
Flan-T5-11b	57.44	59.42 +/- 3.24	61.36 +/- 12.17	48.48 +/- 12.49
Cohere-command-6b	65.00	71.74 +/- 6.99	71.59 +/- 3.15	36.36 +/- 0.00
Cohere-command-52b	72.83	83.33 +/- 3.90	83.33 +/- 2.51	71.21 +/- 6.25
text-ada-001-unknown	57.36	60.87 +/- 7.10	67.80 +/- 3.81	66.67 +/- 6.78
text-babbage-001-unknown	63.89	68.84 +/- 3.90	76.89 +/- 2.43	50.00 +/- 11.44
text-curie-001-unknown	64.39	66.67 +/- 5.98	68.56 +/- 9.94	56.06 +/- 6.25
text-davinci-001-unknown	72.72	93.48 +/- 4.16	80.68 +/- 2.18	57.58 +/- 12.49
text-davinci-002-unknown	75.61	91.30 +/- 2.51	87.12 +/- 2.51	56.06 +/- 8.16
text-davinci-003-unknown	74.31	90.58 +/- 5.28	82.20 +/- 1.56	54.55 +/- 7.42
ChatGPT-unknown	75.11	86.23 +/- 2.99	85.61 +/- 3.12	56.06 +/- 14.29
GPT-4-unknown	82.31	97.10 +/- 3.24	88.64 +/- 3.94	89.39 +/- 3.39
Humans	86.23	93.04	92.73	92.73

Table D.20: Accuracy per label for 1-shot evaluation.

Model	Mean	Particularised	Generalised	Other
OPT-125m	52.72	48.30 +/- 1.83	60.87 +/- 13.04	52.89 +/- 1.04
OPT-350m	52.92	47.73 +/- 2.37	60.87 +/- 11.71	54.35 +/- 3.10
OPT-1.3b	56.31	53.41 +/- 2.71	52.17 +/- 9.88	57.41 +/- 1.38
OPT-2.7b	56.83	49.81 +/- 5.31	69.93 +/- 6.33	55.17 +/- 3.94
OPT-6.7b	60.08	52.65 +/- 7.44	73.55 +/- 3.18	59.09 +/- 5.85
OPT-13b	60.56	53.03 +/- 1.82	71.01 +/- 7.60	59.56 +/- 2.75
OPT-30b	60.33	55.87 +/- 3.31	70.65 +/- 8.01	59.26 +/- 4.61
OPT-66b	63.19	60.04 +/- 4.12	67.39 +/- 8.70	63.39 +/- 4.28
OPT-175b	58.36	56.63 +/- 3.96	59.42 +/- 7.06	57.28 +/- 7.30
BLOOM-560m	54.83	43.94 +/- 3.12	66.30 +/- 8.21	54.82 +/- 1.85
BLOOM-1b1	52.56	47.35 +/- 3.12	63.04 +/- 12.36	51.16 +/- 1.54
BLOOM-1b7	52.81	45.64 +/- 3.50	67.03 +/- 9.92	51.29 +/- 1.44
BLOOM-3b	55.94	45.27 +/- 1.21	76.09 +/- 1.26	55.12 +/- 1.93
BLOOM-7b1	57.00	49.62 +/- 4.08	77.17 +/- 1.66	55.56 +/- 3.57
BLOOM-176b	61.11	58.14 +/- 3.31	66.67 +/- 5.98	59.73 +/- 3.66
EleutherAI-125m	51.67	50.19 +/- 2.49	53.99 +/- 8.82	52.11 +/- 0.89
EleutherAI-1.3b	55.72	50.57 +/- 4.67	57.97 +/- 13.44	57.36 +/- 2.66
EleutherAI-2.7b	55.50	48.67 +/- 4.84	65.22 +/- 4.86	54.22 +/- 2.79
EleutherAI-6b	54.97	49.81 +/- 1.21	66.30 +/- 4.82	54.01 +/- 3.36
EleutherAI-20b	55.86	53.03 +/- 3.57	64.49 +/- 6.36	53.83 +/- 2.41
Cohere-409m	51.89	52.84 +/- 3.98	48.55 +/- 3.69	52.54 +/- 1.96
Cohere-6b	57.86	44.13 +/- 1.66	77.54 +/- 2.05	57.15 +/- 5.08
Cohere-13b	61.78	53.98 +/- 2.05	74.28 +/- 3.64	61.41 +/- 1.84
Cohere-52b	62.97	60.42 +/- 8.18	69.20 +/- 5.24	61.76 +/- 4.21
GPT-3-350m	55.97	50.76 +/- 1.82	73.91 +/- 7.94	54.31 +/- 1.92
GPT-3-1.3b	60.75	53.79 +/- 2.98	68.48 +/- 3.26	61.07 +/- 1.82
GPT-3-6.7b	61.17	55.49 +/- 6.83	72.10 +/- 2.64	60.29 +/- 4.09
GPT-3-175b	65.72	62.31 +/- 4.17	64.86 +/- 7.26	65.33 +/- 2.00
T0-3b	48.89	56.25 +/- 1.57	34.06 +/- 4.29	49.83 +/- 0.55
T0-11b	47.78	56.44 +/- 0.54	27.54 +/- 1.02	49.22 +/- 0.53
BlenderBot-90m	49.94	52.46 +/- 4.27	44.57 +/- 15.66	50.00 +/- 1.65
BlenderBot-3b	53.31	44.51 +/- 0.42	76.09 +/- 0.00	51.59 +/- 0.24
BlenderBot-9b	52.53	54.92 +/- 3.45	55.80 +/- 12.90	50.90 +/- 2.60
Flan-T5-780m	62.89	56.44 +/- 3.32	68.84 +/- 12.90	63.44 +/- 6.28
Flan-T5-3b	52.75	55.11 +/- 1.57	44.20 +/- 5.98	52.41 +/- 3.23
Flan-T5-11b	57.44	53.98 +/- 1.94	62.68 +/- 15.85	57.28 +/- 4.79
Cohere-command-6b	65.00	60.61 +/- 3.69	68.12 +/- 9.53	65.25 +/- 1.37
Cohere-command-52b	72.83	67.42 +/- 2.83	80.07 +/- 2.92	71.36 +/- 1.70
text-ada-001-unknown	57.36	46.97 +/- 2.76	74.64 +/- 2.99	55.90 +/- 3.11
text-babbage-001-unknown	63.89	58.52 +/- 1.43	63.41 +/- 7.26	63.70 +/- 1.10
text-curie-001-unknown	64.39	60.98 +/- 2.14	69.93 +/- 3.42	64.04 +/- 5.79
text-davinci-001-unknown	72.72	62.31 +/- 1.66	76.81 +/- 2.71	72.83 +/- 1.70
text-davinci-002-unknown	75.61	68.18 +/- 2.86	77.54 +/- 2.05	75.32 +/- 3.14
text-davinci-003-unknown	74.31	64.20 +/- 1.43	80.43 +/- 5.02	74.50 +/- 1.29
ChatGPT-unknown	75.11	70.08 +/- 4.38	78.99 +/- 7.50	74.46 +/- 1.19
GPT-4-unknown	82.31	74.43 +/- 2.60	86.96 +/- 3.32	81.70 +/- 1.94
Humans	86.23	83.18	92.17	84.86

Table D.21: Accuracy per label for 5-shot evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
OPT-125m	50.22	44.93 +/- 3.24	57.58 +/- 7.73	57.58 +/- 4.29
OPT-350m	51.47	53.62 +/- 4.81	58.71 +/- 1.56	45.45 +/- 0.00
OPT-1.3b	58.03	68.84 +/- 8.48	63.26 +/- 6.21	30.30 +/- 8.57
OPT-2.7b	57.33	57.97 +/- 4.81	66.67 +/- 4.67	71.21 +/- 3.39
OPT-6.7b	63.31	66.67 +/- 16.20	67.42 +/- 4.08	42.42 +/- 16.32
OPT-13b	67.39	80.43 +/- 4.86	68.94 +/- 4.29	39.39 +/- 6.78
OPT-30b	65.64	84.78 +/- 8.60	66.29 +/- 8.13	37.88 +/- 6.25
OPT-66b	61.50	75.36 +/- 8.20	55.30 +/- 6.90	36.36 +/- 7.42
OPT-175b	63.89	78.26 +/- 7.10	65.15 +/- 2.83	43.94 +/- 3.39
BLOOM-560m	53.75	44.20 +/- 2.99	65.91 +/- 3.94	54.55 +/- 5.25
BLOOM-1b1	57.39	49.28 +/- 6.95	65.15 +/- 4.85	66.67 +/- 6.78
BLOOM-1b7	54.44	61.59 +/- 5.84	56.06 +/- 1.69	43.94 +/- 6.25
BLOOM-3b	57.19	50.72 +/- 3.24	64.77 +/- 4.87	63.64 +/- 12.86
BLOOM-7b1	54.50	50.00 +/- 2.17	62.88 +/- 1.69	69.70 +/- 4.29
BLOOM-176b	65.42	76.09 +/- 6.02	69.32 +/- 4.87	43.94 +/- 3.39
EleutherAI-125m	49.56	50.00 +/- 3.32	50.38 +/- 4.43	34.85 +/- 3.39
EleutherAI-1.3b	57.11	55.07 +/- 5.98	63.64 +/- 4.15	37.88 +/- 11.03
EleutherAI-2.7b	58.03	71.74 +/- 4.16	59.85 +/- 3.12	43.94 +/- 14.29
EleutherAI-6b	58.39	67.39 +/- 6.99	56.82 +/- 6.01	42.42 +/- 18.68
EleutherAI-20b	61.14	65.22 +/- 8.70	64.77 +/- 11.11	30.30 +/- 8.57
Cohere-409m	53.39	47.83 +/- 5.61	59.47 +/- 5.32	31.82 +/- 6.94
Cohere-6b	60.89	65.94 +/- 8.48	66.67 +/- 10.05	45.45 +/- 9.09
Cohere-13b	62.47	81.88 +/- 8.10	62.88 +/- 10.71	34.85 +/- 11.03
Cohere-52b	65.14	73.91 +/- 5.61	67.80 +/- 3.05	51.52 +/- 6.78
GPT-3-350m	55.72	46.38 +/- 3.24	65.53 +/- 1.56	51.52 +/- 4.29
GPT-3-1.3b	62.64	72.46 +/- 10.55	69.70 +/- 4.48	37.88 +/- 12.22
GPT-3-6.7b	62.39	76.81 +/- 14.57	62.50 +/- 5.53	36.36 +/- 7.42
GPT-3-175b	68.72	82.61 +/- 4.35	71.59 +/- 2.54	60.61 +/- 13.55
T0-3b	46.67	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
T0-11b	47.00	52.17 +/- 0.00	39.02 +/- 0.85	36.36 +/- 0.00
BlenderBot-90m	46.58	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-3b	53.36	47.83 +/- 0.00	61.36 +/- 0.00	63.64 +/- 0.00
BlenderBot-9b	52.81	47.83 +/- 4.35	60.98 +/- 0.85	63.64 +/- 0.00
Flan-T5-780m	61.03	61.59 +/- 4.64	70.08 +/- 9.59	42.42 +/- 4.29
Flan-T5-3b	54.89	62.32 +/- 7.39	60.61 +/- 8.26	34.85 +/- 3.39
Flan-T5-11b	61.64	68.84 +/- 6.84	67.80 +/- 8.03	43.94 +/- 8.16
Cohere-command-6b	68.56	77.54 +/- 9.86	78.79 +/- 5.36	39.39 +/- 4.29
Cohere-command-52b	75.42	87.68 +/- 3.90	84.09 +/- 1.31	74.24 +/- 9.70
text-ada-001-unknown	57.61	52.17 +/- 3.55	64.39 +/- 2.83	62.12 +/- 8.16
text-babbage-001-unknown	66.14	71.74 +/- 2.17	77.65 +/- 5.15	57.58 +/- 12.49
text-curie-001-unknown	71.33	76.09 +/- 2.17	70.08 +/- 6.07	43.94 +/- 3.39
text-davinci-001-unknown	74.53	88.41 +/- 3.24	78.03 +/- 5.97	66.67 +/- 12.49
text-davinci-002-unknown	79.56	90.58 +/- 1.62	89.02 +/- 2.04	69.70 +/- 6.78
text-davinci-003-unknown	79.67	89.13 +/- 2.17	86.36 +/- 2.27	74.24 +/- 11.03
ChatGPT-unknown	73.89	86.96 +/- 6.15	87.88 +/- 4.85	75.76 +/- 12.49
GPT-4-unknown	82.03	95.65 +/- 2.51	86.74 +/- 2.04	87.88 +/- 6.78
Humans	86.23	93.04	92.73	92.73

Table D.22: Accuracy per label for 5-shot evaluation.

Model	Mean	Particularised	Generalised	Other
OPT-125m	50.22	47.35 +/- 3.12	60.87 +/- 13.28	48.84 +/- 3.71
OPT-350m	51.47	39.58 +/- 1.79	67.39 +/- 3.07	51.38 +/- 0.95
OPT-1.3b	58.03	56.06 +/- 3.63	57.61 +/- 4.12	58.01 +/- 2.84
OPT-2.7b	57.33	47.35 +/- 3.05	72.46 +/- 2.40	56.20 +/- 3.60
OPT-6.7b	63.31	56.63 +/- 6.93	71.01 +/- 6.48	63.74 +/- 3.18
OPT-13b	67.39	60.23 +/- 2.93	64.86 +/- 2.92	69.12 +/- 2.83
OPT-30b	65.64	59.85 +/- 1.42	59.42 +/- 7.70	67.27 +/- 4.36
OPT-66b	61.50	56.44 +/- 3.51	58.70 +/- 9.64	63.65 +/- 3.93
OPT-175b	63.89	61.55 +/- 2.22	52.54 +/- 5.53	65.33 +/- 2.44
BLOOM-560m	53.75	44.89 +/- 2.05	73.19 +/- 6.11	52.50 +/- 0.78
BLOOM-1b1	57.39	48.67 +/- 3.44	70.65 +/- 4.12	57.02 +/- 1.86
BLOOM-1b7	54.44	48.86 +/- 4.91	60.14 +/- 3.24	54.74 +/- 0.96
BLOOM-3b	57.19	50.00 +/- 3.35	72.46 +/- 2.40	56.24 +/- 0.89
BLOOM-7b1	54.50	46.02 +/- 2.25	72.10 +/- 4.24	53.10 +/- 1.04
BLOOM-176b	65.42	65.53 +/- 4.38	49.28 +/- 9.69	66.88 +/- 3.42
EleutherAI-125m	49.56	44.32 +/- 5.76	56.88 +/- 4.05	50.04 +/- 2.49
EleutherAI-1.3b	57.11	50.76 +/- 3.26	69.93 +/- 4.77	56.89 +/- 1.57
EleutherAI-2.7b	58.03	51.52 +/- 2.43	61.59 +/- 4.10	58.61 +/- 1.39
EleutherAI-6b	58.39	49.62 +/- 1.69	63.04 +/- 5.02	59.91 +/- 5.04
EleutherAI-20b	61.14	51.52 +/- 2.43	61.59 +/- 12.78	63.48 +/- 5.29
Cohere-409m	53.39	50.38 +/- 1.69	68.48 +/- 4.82	52.45 +/- 0.99
Cohere-6b	60.89	52.65 +/- 2.14	64.49 +/- 5.98	61.80 +/- 4.77
Cohere-13b	62.47	59.66 +/- 6.11	68.84 +/- 7.28	62.02 +/- 3.56
Cohere-52b	65.14	60.04 +/- 4.07	68.12 +/- 7.39	65.46 +/- 3.30
GPT-3-350m	55.72	44.70 +/- 1.26	74.28 +/- 6.07	55.47 +/- 1.59
GPT-3-1.3b	62.64	49.24 +/- 2.51	67.39 +/- 4.35	64.38 +/- 2.51
GPT-3-6.7b	62.39	51.70 +/- 2.25	64.86 +/- 7.68	64.38 +/- 2.30
GPT-3-175b	68.72	60.98 +/- 5.74	66.67 +/- 8.76	69.90 +/- 0.68
T0-3b	46.67	55.68 +/- 0.00	23.91 +/- 0.00	48.32 +/- 0.15
T0-11b	47.00	55.87 +/- 0.42	25.00 +/- 1.66	48.62 +/- 0.23
BlenderBot-90m	46.58	55.11 +/- 1.27	24.28 +/- 0.81	48.28 +/- 0.31
BlenderBot-3b	53.36	44.32 +/- 0.00	76.09 +/- 0.00	51.72 +/- 0.10
BlenderBot-9b	52.81	44.32 +/- 0.93	75.72 +/- 0.81	50.95 +/- 1.02
Flan-T5-780m	61.03	54.36 +/- 3.50	71.01 +/- 12.96	60.77 +/- 6.09
Flan-T5-3b	54.89	57.01 +/- 1.79	41.30 +/- 9.64	55.47 +/- 4.00
Flan-T5-11b	61.64	56.25 +/- 3.20	64.86 +/- 17.04	61.76 +/- 5.21
Cohere-command-6b	68.56	60.23 +/- 5.00	74.28 +/- 6.57	68.91 +/- 1.47
Cohere-command-52b	75.42	70.08 +/- 3.39	77.17 +/- 3.26	74.68 +/- 2.80
text-ada-001-unknown	57.61	48.86 +/- 2.18	72.83 +/- 1.66	57.11 +/- 3.74
text-babbage-001-unknown	66.14	57.01 +/- 2.82	71.74 +/- 2.81	66.06 +/- 0.85
text-curie-001-unknown	71.33	60.04 +/- 0.78	69.93 +/- 5.67	74.63 +/- 0.98
text-davinci-001-unknown	74.53	60.80 +/- 3.75	81.16 +/- 3.69	75.80 +/- 1.32
text-davinci-002-unknown	79.56	71.02 +/- 2.76	87.68 +/- 1.62	79.03 +/- 2.26
text-davinci-003-unknown	79.67	71.59 +/- 1.86	87.68 +/- 1.02	79.33 +/- 1.12
ChatGPT-unknown	73.89	69.51 +/- 4.80	73.91 +/- 11.64	72.44 +/- 6.16
GPT-4-unknown	82.03	71.21 +/- 2.91	87.32 +/- 3.64	82.30 +/- 2.31
Humans	86.23	83.18	92.17	84.86

Table D.23: Accuracy per label for 10-shot evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
OPT-125m	52.89	55.80 +/- 8.48	55.68 +/- 8.78	66.67 +/- 6.78
OPT-350m	56.72	57.97 +/- 4.10	60.61 +/- 3.86	65.15 +/- 16.11
OPT-1.3b	59.92	70.29 +/- 4.64	54.17 +/- 3.57	34.85 +/- 3.39
OPT-2.7b	58.03	52.17 +/- 2.51	65.53 +/- 2.76	63.64 +/- 5.25
OPT-6.7b	63.28	71.01 +/- 5.42	65.53 +/- 6.07	53.03 +/- 17.73
OPT-13b	65.75	77.54 +/- 6.84	63.26 +/- 4.23	50.00 +/- 10.16
OPT-30b	63.36	78.99 +/- 4.64	56.06 +/- 8.57	33.33 +/- 4.29
OPT-66b	60.81	71.01 +/- 7.80	54.55 +/- 9.46	36.36 +/- 0.00
OPT-175b	60.75	76.81 +/- 4.10	58.33 +/- 6.52	43.94 +/- 11.03
BLOOM-560m	54.56	49.28 +/- 3.24	64.39 +/- 3.12	63.64 +/- 0.00
BLOOM-1b1	57.31	55.07 +/- 5.42	60.61 +/- 5.36	59.09 +/- 8.70
BLOOM-1b7	53.14	68.12 +/- 6.48	45.45 +/- 2.62	59.09 +/- 11.44
BLOOM-3b	59.39	54.35 +/- 4.86	65.53 +/- 5.32	66.67 +/- 4.29
BLOOM-7b1	56.11	53.62 +/- 7.39	65.53 +/- 3.81	69.70 +/- 6.78
BLOOM-176b	63.47	75.36 +/- 5.98	68.18 +/- 9.00	42.42 +/- 4.29
EleutherAI-125m	54.39	58.70 +/- 6.02	52.27 +/- 5.41	83.33 +/- 19.93
EleutherAI-1.3b	57.83	67.39 +/- 5.47	60.61 +/- 5.19	62.12 +/- 11.03
EleutherAI-2.7b	57.03	73.19 +/- 2.99	55.68 +/- 5.37	66.67 +/- 13.55
EleutherAI-6b	57.64	64.49 +/- 3.90	51.14 +/- 11.04	56.06 +/- 16.94
EleutherAI-20b	59.33	67.39 +/- 5.47	62.12 +/- 10.47	37.88 +/- 6.25
Cohere-409m	53.92	63.04 +/- 5.47	46.21 +/- 6.90	51.52 +/- 11.34
Cohere-6b	58.72	66.67 +/- 9.61	63.26 +/- 14.46	50.00 +/- 12.59
Cohere-13b	60.36	76.81 +/- 6.48	56.06 +/- 9.52	34.85 +/- 3.39
Cohere-52b	63.31	72.46 +/- 5.98	68.18 +/- 4.55	51.52 +/- 10.05
GPT-3-350m	57.72	53.62 +/- 3.24	63.64 +/- 6.01	65.15 +/- 8.16
GPT-3-1.3b	60.92	73.19 +/- 7.28	59.47 +/- 5.78	48.48 +/- 8.57
GPT-3-6.7b	63.94	71.01 +/- 7.80	67.80 +/- 1.56	40.91 +/- 8.70
GPT-3-175b	67.28	76.81 +/- 9.61	68.56 +/- 5.32	81.82 +/- 10.50
T0-3b	46.67	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
T0-11b	46.72	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-90m	46.67	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-3b	53.25	47.83 +/- 0.00	61.36 +/- 0.00	63.64 +/- 0.00
BlenderBot-9b	53.36	42.03 +/- 4.10	62.12 +/- 3.12	63.64 +/- 0.00
Flan-T5-780m	60.19	63.04 +/- 2.17	68.56 +/- 10.77	40.91 +/- 4.55
Flan-T5-3b	55.14	61.59 +/- 6.84	58.71 +/- 10.37	36.36 +/- 0.00
Flan-T5-11b	60.56	67.39 +/- 7.43	70.83 +/- 10.45	40.91 +/- 4.55
Cohere-command-6b	68.22	78.99 +/- 5.28	74.62 +/- 5.32	36.36 +/- 0.00
Cohere-command-52b	75.64	88.41 +/- 3.24	84.85 +/- 2.51	66.67 +/- 8.57
text-ada-001-unknown	57.36	64.49 +/- 7.28	56.44 +/- 5.78	57.58 +/- 6.78
text-babbage-001-unknown	63.53	67.39 +/- 2.17	73.11 +/- 4.62	68.18 +/- 4.55
text-curie-001-unknown	70.17	83.33 +/- 1.62	76.14 +/- 2.86	45.45 +/- 5.25
text-davinci-001-unknown	74.97	89.13 +/- 2.17	83.33 +/- 2.83	59.09 +/- 6.94
text-davinci-002-unknown	79.56	93.48 +/- 2.17	88.26 +/- 0.85	66.67 +/- 8.57
text-davinci-003-unknown	79.00	94.93 +/- 3.90	85.61 +/- 3.39	66.67 +/- 4.29
ChatGPT-unknown	74.28	84.06 +/- 6.48	86.36 +/- 4.35	62.12 +/- 11.03
GPT-4-unknown	81.31	94.93 +/- 2.99	86.74 +/- 4.03	89.39 +/- 3.39
Humans	86.23	93.04	92.73	92.73

Table D.24: Accuracy per label for 10-shot evaluation.

Model	Mean	Particularised	Generalised	Other
OPT-125m	52.89	51.33 +/- 4.52	57.97 +/- 15.30	51.68 +/- 1.58
OPT-350m	56.72	55.11 +/- 1.57	72.10 +/- 2.32	54.39 +/- 1.50
OPT-1.3b	59.92	60.23 +/- 0.93	61.23 +/- 2.32	60.34 +/- 2.77
OPT-2.7b	58.03	49.62 +/- 3.32	74.64 +/- 2.05	57.19 +/- 3.77
OPT-6.7b	63.28	58.33 +/- 3.97	73.19 +/- 3.48	62.70 +/- 3.22
OPT-13b	65.75	60.23 +/- 2.45	72.10 +/- 3.18	66.37 +/- 2.84
OPT-30b	63.36	62.31 +/- 2.40	65.22 +/- 6.28	64.17 +/- 3.73
OPT-66b	60.81	57.58 +/- 1.26	60.51 +/- 7.98	62.36 +/- 3.33
OPT-175b	60.75	60.98 +/- 2.24	56.16 +/- 2.92	60.94 +/- 3.42
BLOOM-560m	54.56	46.21 +/- 1.26	73.19 +/- 2.99	53.06 +/- 0.77
BLOOM-1b1	57.31	47.54 +/- 4.97	64.13 +/- 7.40	58.31 +/- 3.30
BLOOM-1b7	53.14	53.03 +/- 3.45	53.62 +/- 4.64	52.80 +/- 1.74
BLOOM-3b	59.39	55.49 +/- 2.22	69.57 +/- 5.89	58.35 +/- 0.41
BLOOM-7b1	56.11	49.62 +/- 5.07	71.38 +/- 2.92	54.35 +/- 3.34
BLOOM-176b	63.47	68.18 +/- 3.47	50.36 +/- 9.00	63.35 +/- 4.15
EleutherAI-125m	54.39	55.11 +/- 3.13	63.41 +/- 6.20	52.20 +/- 2.23
EleutherAI-1.3b	57.83	50.00 +/- 2.78	69.20 +/- 2.32	57.15 +/- 1.52
EleutherAI-2.7b	57.03	57.20 +/- 2.76	57.25 +/- 6.23	55.77 +/- 0.96
EleutherAI-6b	57.64	56.25 +/- 2.76	59.42 +/- 10.25	58.05 +/- 4.71
EleutherAI-20b	59.33	57.39 +/- 1.83	63.04 +/- 13.63	59.04 +/- 3.30
Cohere-409m	53.92	57.39 +/- 2.60	66.30 +/- 4.98	51.94 +/- 1.99
Cohere-6b	58.72	51.70 +/- 2.34	64.86 +/- 6.33	58.74 +/- 5.75
Cohere-13b	60.36	58.52 +/- 1.27	70.29 +/- 5.98	59.73 +/- 5.24
Cohere-52b	63.31	53.41 +/- 2.93	67.75 +/- 7.88	64.17 +/- 2.12
GPT-3-350m	57.72	50.95 +/- 2.89	73.91 +/- 1.26	56.59 +/- 1.79
GPT-3-1.3b	60.92	57.01 +/- 5.01	63.77 +/- 2.71	61.15 +/- 1.10
GPT-3-6.7b	63.94	58.52 +/- 4.19	66.67 +/- 5.28	64.56 +/- 1.31
GPT-3-175b	67.28	63.45 +/- 2.66	68.84 +/- 4.64	66.75 +/- 2.65
T0-3b	46.67	55.68 +/- 0.00	24.28 +/- 0.81	48.28 +/- 0.10
T0-11b	46.72	55.68 +/- 0.00	24.28 +/- 0.81	48.36 +/- 0.10
BlenderBot-90m	46.67	55.68 +/- 0.00	23.91 +/- 0.00	48.32 +/- 0.00
BlenderBot-3b	53.25	44.70 +/- 0.54	76.09 +/- 0.00	51.46 +/- 0.23
BlenderBot-9b	53.36	45.27 +/- 1.21	76.09 +/- 1.77	51.77 +/- 0.69
Flan-T5-780m	60.19	54.17 +/- 2.51	71.38 +/- 10.75	59.60 +/- 4.49
Flan-T5-3b	55.14	54.92 +/- 1.42	43.48 +/- 12.10	56.29 +/- 3.84
Flan-T5-11b	60.56	59.66 +/- 3.33	57.61 +/- 13.09	60.03 +/- 5.13
Cohere-command-6b	68.22	63.07 +/- 4.44	77.17 +/- 6.73	67.79 +/- 2.45
Cohere-command-52b	75.64	70.27 +/- 1.53	76.45 +/- 4.24	75.15 +/- 1.17
text-ada-001-unknown	57.36	49.24 +/- 3.32	61.96 +/- 5.14	58.23 +/- 1.53
text-babbage-001-unknown	63.53	56.63 +/- 2.22	65.22 +/- 5.47	63.35 +/- 1.49
text-curie-001-unknown	70.17	62.69 +/- 2.12	67.75 +/- 7.36	71.32 +/- 1.01
text-davinci-001-unknown	74.97	63.83 +/- 1.21	80.80 +/- 1.95	75.41 +/- 1.79
text-davinci-002-unknown	79.56	70.08 +/- 1.56	84.78 +/- 2.51	79.59 +/- 2.79
text-davinci-003-unknown	79.00	68.18 +/- 1.31	87.32 +/- 1.49	79.07 +/- 1.38
ChatGPT-unknown	74.28	68.37 +/- 4.37	75.36 +/- 11.06	73.90 +/- 4.82
GPT-4-unknown	81.31	70.83 +/- 4.29	86.96 +/- 2.81	81.31 +/- 3.82
Humans	86.23	83.18	92.17	84.86

Table D.25: Accuracy per label for 15-shot evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
OPT-125m	51.86	44.93 +/- 4.10	53.41 +/- 8.48	43.94 +/- 19.93
OPT-350m	55.42	48.55 +/- 2.99	48.48 +/- 2.51	42.42 +/- 6.78
OPT-1.3b	61.61	64.49 +/- 5.28	68.94 +/- 4.67	42.42 +/- 6.78
OPT-2.7b	59.53	55.80 +/- 5.84	62.50 +/- 2.18	60.61 +/- 4.29
OPT-6.7b	64.72	55.80 +/- 7.28	68.18 +/- 3.47	60.61 +/- 16.32
OPT-13b	65.17	64.49 +/- 6.36	66.67 +/- 6.11	54.55 +/- 5.25
OPT-30b	64.06	68.84 +/- 4.64	60.23 +/- 5.98	43.94 +/- 8.16
OPT-66b	61.83	65.94 +/- 11.34	55.30 +/- 8.26	39.39 +/- 4.29
OPT-175b	64.78	76.09 +/- 11.16	67.05 +/- 9.44	50.00 +/- 6.94
BLOOM-560m	55.00	47.83 +/- 2.51	59.09 +/- 2.27	62.12 +/- 3.39
BLOOM-1b1	57.58	50.00 +/- 4.86	53.03 +/- 2.51	57.58 +/- 4.29
BLOOM-1b7	55.14	60.14 +/- 12.40	50.38 +/- 4.98	53.03 +/- 16.94
BLOOM-3b	58.69	44.93 +/- 3.24	61.36 +/- 6.94	57.58 +/- 6.78
BLOOM-7b1	55.67	55.07 +/- 7.80	61.36 +/- 2.93	56.06 +/- 8.16
BLOOM-176b	61.89	77.54 +/- 9.86	70.08 +/- 7.59	37.88 +/- 3.39
EleutherAI-125m	56.03	60.14 +/- 7.70	42.80 +/- 4.62	59.09 +/- 13.64
EleutherAI-1.3b	57.44	49.28 +/- 2.05	51.52 +/- 6.11	39.39 +/- 15.45
EleutherAI-2.7b	58.08	53.62 +/- 4.81	57.20 +/- 4.81	56.06 +/- 11.03
EleutherAI-6b	58.81	58.70 +/- 10.87	56.06 +/- 10.47	56.06 +/- 6.25
EleutherAI-20b	59.86	55.80 +/- 2.99	63.64 +/- 9.19	42.42 +/- 4.29
Cohere-409m	55.19	50.00 +/- 4.16	50.76 +/- 4.85	42.42 +/- 6.78
Cohere-6b	60.44	65.94 +/- 9.19	67.05 +/- 9.88	50.00 +/- 17.99
Cohere-13b	62.83	67.39 +/- 13.92	64.77 +/- 5.53	43.94 +/- 9.70
Cohere-52b	64.72	63.04 +/- 6.52	69.32 +/- 7.28	63.64 +/- 13.89
GPT-3-350m	58.83	50.00 +/- 7.43	53.41 +/- 1.74	42.42 +/- 13.55
GPT-3-1.3b	62.86	53.62 +/- 7.39	65.91 +/- 3.71	50.00 +/- 14.61
GPT-3-6.7b	65.17	62.32 +/- 6.95	63.64 +/- 2.27	51.52 +/- 10.05
GPT-3-175b	68.31	78.26 +/- 5.02	66.67 +/- 4.48	56.06 +/- 11.03
T0-3b	46.67	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
T0-11b	46.81	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-90m	46.56	52.17 +/- 0.00	38.64 +/- 0.00	34.85 +/- 3.39
BlenderBot-3b	53.14	47.83 +/- 0.00	61.36 +/- 0.00	63.64 +/- 0.00
BlenderBot-9b	53.19	45.65 +/- 3.32	60.61 +/- 5.03	65.15 +/- 3.39
Flan-T5-780m	61.50	65.94 +/- 5.84	67.42 +/- 10.71	42.42 +/- 4.29
Flan-T5-3b	55.08	66.67 +/- 10.55	60.23 +/- 11.64	36.36 +/- 0.00
Flan-T5-11b	60.83	65.94 +/- 7.28	68.56 +/- 8.65	45.45 +/- 7.42
Cohere-command-6b	70.03	80.43 +/- 3.32	78.41 +/- 2.18	45.45 +/- 10.50
Cohere-command-52b	75.39	89.13 +/- 2.17	83.33 +/- 1.69	72.73 +/- 5.25
text-ada-001-unknown	58.28	55.07 +/- 5.98	56.06 +/- 5.67	63.64 +/- 13.89
text-babbage-001-unknown	65.19	63.04 +/- 4.86	77.27 +/- 3.71	68.18 +/- 6.94
text-curie-001-unknown	69.92	79.71 +/- 2.05	73.11 +/- 1.56	45.45 +/- 10.50
text-davinci-001-unknown	75.31	88.41 +/- 2.05	82.95 +/- 2.86	57.58 +/- 8.57
text-davinci-002-unknown	79.06	94.93 +/- 1.62	85.23 +/- 2.86	72.73 +/- 15.75
text-davinci-003-unknown	79.03	91.30 +/- 0.00	85.61 +/- 1.69	69.70 +/- 15.45
ChatGPT-unknown	75.56	86.23 +/- 4.64	86.74 +/- 4.03	60.61 +/- 10.05
GPT-4-unknown	82.08	95.65 +/- 2.51	81.44 +/- 2.43	90.91 +/- 0.00
Humans	86.23	93.04	92.73	92.73

Table D.26: Accuracy per label for 15-shot evaluation.

Model	Mean	Particularised	Generalised	Other
OPT-125m	51.86	50.95 +/- 4.32	55.80 +/- 20.34	51.94 +/- 0.84
OPT-350m	55.42	55.30 +/- 3.39	60.51 +/- 4.24	56.29 +/- 0.75
OPT-1.3b	61.61	57.39 +/- 5.90	63.77 +/- 6.60	61.76 +/- 3.50
OPT-2.7b	59.53	49.24 +/- 2.98	75.00 +/- 1.66	59.78 +/- 5.03
OPT-6.7b	64.72	58.71 +/- 7.78	74.28 +/- 5.24	65.12 +/- 2.69
OPT-13b	65.17	64.02 +/- 2.34	65.22 +/- 7.32	65.50 +/- 1.25
OPT-30b	64.06	62.50 +/- 3.99	61.59 +/- 6.84	65.42 +/- 3.99
OPT-66b	61.83	58.71 +/- 7.21	57.61 +/- 5.14	64.25 +/- 3.20
OPT-175b	64.78	64.20 +/- 4.03	59.78 +/- 3.92	64.90 +/- 3.66
BLOOM-560m	55.00	44.13 +/- 2.58	74.64 +/- 4.64	54.78 +/- 1.75
BLOOM-1b1	57.58	45.45 +/- 1.74	64.13 +/- 5.58	60.42 +/- 2.34
BLOOM-1b7	55.14	52.27 +/- 5.21	57.61 +/- 8.40	55.73 +/- 0.84
BLOOM-3b	58.69	49.81 +/- 1.21	74.28 +/- 2.92	59.30 +/- 0.95
BLOOM-7b1	55.67	48.67 +/- 3.74	70.29 +/- 3.90	54.78 +/- 3.23
BLOOM-176b	61.89	64.02 +/- 4.43	46.38 +/- 11.20	62.02 +/- 4.53
EleutherAI-125m	56.03	56.82 +/- 3.47	54.35 +/- 5.89	57.11 +/- 0.65
EleutherAI-1.3b	57.44	51.14 +/- 3.21	61.59 +/- 9.11	59.95 +/- 2.44
EleutherAI-2.7b	58.08	57.58 +/- 2.24	61.96 +/- 6.49	58.27 +/- 1.50
EleutherAI-6b	58.81	54.17 +/- 6.07	66.67 +/- 8.48	59.35 +/- 4.11
EleutherAI-20b	59.86	55.11 +/- 3.98	62.68 +/- 11.87	60.85 +/- 4.53
Cohere-409m	55.19	52.65 +/- 1.69	61.23 +/- 8.08	56.12 +/- 2.19
Cohere-6b	60.44	49.62 +/- 2.60	71.01 +/- 6.48	60.85 +/- 4.38
Cohere-13b	62.83	57.77 +/- 3.17	71.01 +/- 5.98	63.09 +/- 2.62
Cohere-52b	64.72	57.39 +/- 2.69	71.01 +/- 4.81	65.16 +/- 1.07
GPT-3-350m	58.83	55.68 +/- 2.54	65.22 +/- 8.96	60.29 +/- 1.99
GPT-3-1.3b	62.86	56.06 +/- 5.67	63.04 +/- 7.63	64.86 +/- 1.78
GPT-3-6.7b	65.17	58.33 +/- 4.62	73.55 +/- 7.88	66.37 +/- 2.47
GPT-3-175b	68.31	64.77 +/- 4.10	71.38 +/- 6.33	68.60 +/- 3.97
T0-3b	46.67	55.68 +/- 0.00	23.91 +/- 0.00	48.32 +/- 0.00
T0-11b	46.81	55.68 +/- 0.00	25.00 +/- 1.09	48.41 +/- 0.12
BlenderBot-90m	46.56	55.68 +/- 0.00	23.91 +/- 0.00	48.19 +/- 0.13
BlenderBot-3b	53.14	44.32 +/- 0.00	75.36 +/- 1.02	51.46 +/- 0.23
BlenderBot-9b	53.19	44.13 +/- 1.79	75.72 +/- 1.49	51.72 +/- 0.74
Flan-T5-780m	61.50	56.63 +/- 2.22	71.74 +/- 11.30	60.90 +/- 4.55
Flan-T5-3b	55.08	56.82 +/- 1.31	45.65 +/- 12.92	55.04 +/- 3.39
Flan-T5-11b	60.83	57.01 +/- 3.37	60.14 +/- 15.81	61.02 +/- 5.34
Cohere-command-6b	70.03	60.80 +/- 4.72	72.83 +/- 8.30	70.84 +/- 1.68
Cohere-command-52b	75.39	69.89 +/- 2.43	76.81 +/- 3.90	74.76 +/- 1.01
text-ada-001-unknown	58.28	52.08 +/- 3.74	69.20 +/- 3.42	58.57 +/- 2.04
text-babbage-001-unknown	65.19	58.33 +/- 2.43	67.03 +/- 3.42	65.12 +/- 2.40
text-curie-001-unknown	69.92	62.50 +/- 1.47	68.84 +/- 6.72	71.40 +/- 0.93
text-davinci-001-unknown	75.31	64.58 +/- 2.12	83.70 +/- 1.66	75.54 +/- 0.95
text-davinci-002-unknown	79.06	72.92 +/- 1.02	86.96 +/- 3.97	77.99 +/- 2.77
text-davinci-003-unknown	79.03	69.32 +/- 2.37	87.68 +/- 1.62	78.94 +/- 1.12
ChatGPT-unknown	75.56	72.16 +/- 4.81	77.54 +/- 7.90	74.63 +/- 4.41
GPT-4-unknown	82.08	72.92 +/- 1.02	86.23 +/- 2.99	82.69 +/- 3.87
Humans	86.23	83.18	92.17	84.86

Table D.27: Accuracy per label for 30-shot evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
OPT-125m	51.50	55.80 +/- 5.28	54.55 +/- 9.28	54.55 +/- 9.09
OPT-350m	54.61	49.28 +/- 7.39	56.82 +/- 1.86	37.88 +/- 9.70
OPT-1.3b	61.67	71.01 +/- 3.24	67.80 +/- 5.48	28.79 +/- 11.03
OPT-2.7b	59.86	58.70 +/- 10.87	71.21 +/- 8.37	46.97 +/- 11.03
OPT-6.7b	63.61	62.32 +/- 11.13	67.05 +/- 2.86	46.97 +/- 19.93
OPT-13b	63.39	60.14 +/- 5.28	60.98 +/- 5.93	46.97 +/- 6.25
OPT-30b	65.47	71.74 +/- 8.23	62.88 +/- 7.38	37.88 +/- 3.39
OPT-66b	60.83	60.14 +/- 3.90	51.52 +/- 11.79	43.94 +/- 6.25
OPT-175b	62.44	65.94 +/- 10.77	62.50 +/- 13.43	60.61 +/- 4.29
BLOOM-560m	55.00	47.10 +/- 1.62	60.98 +/- 2.43	62.12 +/- 3.39
BLOOM-1b1	56.89	49.28 +/- 3.24	54.92 +/- 8.65	46.97 +/- 8.16
BLOOM-1b7	52.28	52.90 +/- 7.28	47.35 +/- 9.32	36.36 +/- 16.60
BLOOM-3b	58.64	50.72 +/- 2.05	62.50 +/- 5.68	59.09 +/- 6.94
BLOOM-7b1	57.61	50.72 +/- 5.98	61.74 +/- 3.81	54.55 +/- 9.09
BLOOM-176b	61.06	73.19 +/- 8.85	66.29 +/- 9.41	48.48 +/- 4.29
EleutherAI-125m	53.44	47.10 +/- 4.64	47.73 +/- 6.43	39.39 +/- 15.45
EleutherAI-1.3b	55.97	44.93 +/- 4.81	51.89 +/- 6.74	37.88 +/- 6.25
EleutherAI-2.7b	57.36	62.32 +/- 5.98	53.41 +/- 2.86	37.88 +/- 11.03
EleutherAI-6b	58.75	59.42 +/- 12.21	52.27 +/- 14.43	36.36 +/- 5.25
EleutherAI-20b	57.36	57.97 +/- 5.42	60.61 +/- 10.30	31.82 +/- 8.70
Cohere-409m	57.17	47.83 +/- 3.55	53.41 +/- 3.15	53.03 +/- 3.39
Cohere-6b	60.36	58.70 +/- 13.92	62.50 +/- 10.23	54.55 +/- 10.50
Cohere-13b	64.81	70.29 +/- 21.65	65.91 +/- 7.07	45.45 +/- 5.25
Cohere-52b	65.72	67.39 +/- 8.60	66.29 +/- 1.56	53.03 +/- 11.03
GPT-3-350m	60.25	55.07 +/- 2.05	57.95 +/- 5.83	51.52 +/- 10.05
GPT-3-1.3b	60.19	61.59 +/- 3.90	54.92 +/- 6.99	43.94 +/- 9.70
GPT-3-6.7b	62.86	56.52 +/- 4.35	65.53 +/- 3.32	50.00 +/- 6.94
GPT-3-175b	68.31	67.39 +/- 5.47	72.73 +/- 2.62	75.76 +/- 4.29
T0-3b	46.67	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
T0-11b	46.75	52.17 +/- 0.00	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-90m	46.67	51.45 +/- 1.62	38.64 +/- 0.00	36.36 +/- 0.00
BlenderBot-3b	53.25	47.83 +/- 0.00	61.36 +/- 0.00	63.64 +/- 0.00
BlenderBot-9b	53.72	46.38 +/- 4.10	63.26 +/- 3.32	63.64 +/- 0.00
Flan-T5-780m	61.50	67.39 +/- 8.60	70.83 +/- 6.35	42.42 +/- 4.29
Flan-T5-3b	56.11	65.22 +/- 7.10	62.50 +/- 13.04	36.36 +/- 0.00
Flan-T5-11b	62.11	67.39 +/- 10.27	72.73 +/- 10.66	51.52 +/- 15.45
Cohere-command-6b	70.44	81.16 +/- 3.24	78.03 +/- 2.83	46.97 +/- 8.16
Cohere-command-52b	75.00	85.51 +/- 2.05	78.41 +/- 1.14	78.79 +/- 6.78
text-ada-001-unknown	55.58	50.72 +/- 7.39	57.58 +/- 4.29	57.58 +/- 8.57
text-babbage-001-unknown	66.00	67.39 +/- 5.47	71.59 +/- 3.15	63.64 +/- 5.25
text-curie-001-unknown	70.33	75.36 +/- 3.24	76.52 +/- 5.03	60.61 +/- 8.57
text-davinci-001-unknown	75.83	85.51 +/- 2.05	84.09 +/- 1.86	65.15 +/- 8.16
text-davinci-002-unknown	80.64	97.83 +/- 2.17	87.50 +/- 2.18	83.33 +/- 3.39
text-davinci-003-unknown	79.53	94.93 +/- 1.62	84.85 +/- 3.39	81.82 +/- 9.09
ChatGPT-unknown	75.64	87.68 +/- 4.64	89.02 +/- 4.43	83.33 +/- 9.70
GPT-4-unknown	82.17	95.65 +/- 3.55	87.12 +/- 3.12	90.91 +/- 0.00
Humans	86.23	93.04	92.73	92.73

Table D.28: Accuracy per label for 30-shot evaluation.

Model	Mean	Particularised	Generalised	Other
OPT-125m	51.50	50.38 +/- 4.29	52.17 +/- 23.95	50.99 +/- 2.42
OPT-350m	54.61	55.30 +/- 1.26	49.28 +/- 3.48	55.51 +/- 1.92
OPT-1.3b	61.67	56.25 +/- 2.76	56.16 +/- 4.05	63.14 +/- 5.41
OPT-2.7b	59.86	50.19 +/- 2.89	69.57 +/- 4.16	59.95 +/- 5.23
OPT-6.7b	63.61	58.90 +/- 6.48	72.10 +/- 8.08	63.74 +/- 4.30
OPT-13b	63.39	59.47 +/- 2.14	64.86 +/- 6.07	65.07 +/- 2.24
OPT-30b	65.47	63.64 +/- 3.01	63.04 +/- 7.94	66.93 +/- 5.52
OPT-66b	60.83	63.07 +/- 3.64	55.43 +/- 9.29	62.62 +/- 4.98
OPT-175b	62.44	64.39 +/- 2.34	51.09 +/- 5.72	63.18 +/- 2.85
BLOOM-560m	55.00	47.92 +/- 2.31	71.01 +/- 6.23	54.18 +/- 1.82
BLOOM-1b1	56.89	49.81 +/- 4.93	56.88 +/- 8.46	59.35 +/- 3.16
BLOOM-1b7	52.28	55.49 +/- 1.79	40.94 +/- 8.82	53.83 +/- 1.25
BLOOM-3b	58.64	52.46 +/- 1.53	68.84 +/- 3.69	58.74 +/- 1.44
BLOOM-7b1	57.61	54.36 +/- 8.98	70.29 +/- 2.05	56.76 +/- 4.88
BLOOM-176b	61.06	67.42 +/- 1.93	50.00 +/- 9.05	60.08 +/- 4.01
EleutherAI-125m	53.44	59.47 +/- 2.43	46.74 +/- 4.65	54.18 +/- 1.21
EleutherAI-1.3b	55.97	51.70 +/- 5.20	56.52 +/- 10.80	58.40 +/- 1.35
EleutherAI-2.7b	57.36	57.39 +/- 2.25	51.81 +/- 3.18	58.79 +/- 0.89
EleutherAI-6b	58.75	57.77 +/- 4.22	58.33 +/- 9.92	60.38 +/- 4.52
EleutherAI-20b	57.36	51.52 +/- 4.08	59.42 +/- 14.51	58.79 +/- 2.39
Cohere-409m	57.17	51.14 +/- 3.99	64.49 +/- 8.57	58.66 +/- 2.15
Cohere-6b	60.36	51.52 +/- 2.51	70.29 +/- 6.11	61.20 +/- 5.15
Cohere-13b	64.81	58.14 +/- 2.96	68.12 +/- 8.20	65.98 +/- 3.57
Cohere-52b	65.72	64.96 +/- 4.17	69.93 +/- 2.32	65.50 +/- 2.01
GPT-3-350m	60.25	57.58 +/- 2.51	65.22 +/- 6.28	60.98 +/- 1.63
GPT-3-1.3b	60.19	57.77 +/- 5.66	57.61 +/- 9.12	62.06 +/- 4.02
GPT-3-6.7b	62.86	61.17 +/- 6.41	63.41 +/- 7.98	63.65 +/- 2.52
GPT-3-175b	68.31	64.58 +/- 5.06	70.29 +/- 6.23	68.17 +/- 2.01
T0-3b	46.67	55.68 +/- 0.00	23.91 +/- 0.00	48.32 +/- 0.00
T0-11b	46.75	55.87 +/- 0.42	23.91 +/- 0.00	48.41 +/- 0.12
BlenderBot-90m	46.67	55.68 +/- 0.00	23.55 +/- 0.81	48.41 +/- 0.24
BlenderBot-3b	53.25	44.32 +/- 0.00	76.09 +/- 0.00	51.55 +/- 0.20
BlenderBot-9b	53.72	45.08 +/- 1.82	75.36 +/- 1.02	52.11 +/- 0.93
Flan-T5-780m	61.50	57.01 +/- 3.85	73.19 +/- 8.67	60.16 +/- 3.75
Flan-T5-3b	56.11	56.44 +/- 1.56	47.83 +/- 11.71	56.29 +/- 4.18
Flan-T5-11b	62.11	61.36 +/- 4.50	57.25 +/- 16.68	61.58 +/- 6.25
Cohere-command-6b	70.44	64.96 +/- 1.21	78.62 +/- 5.53	69.81 +/- 1.60
Cohere-command-52b	75.00	71.78 +/- 1.66	75.00 +/- 3.49	74.55 +/- 0.49
text-ada-001-unknown	55.58	53.41 +/- 1.97	56.52 +/- 4.35	55.86 +/- 3.03
text-babbage-001-unknown	66.00	60.80 +/- 2.69	62.32 +/- 6.11	66.88 +/- 2.36
text-curie-001-unknown	70.33	60.42 +/- 5.01	74.28 +/- 6.20	71.32 +/- 1.42
text-davinci-001-unknown	75.83	67.05 +/- 1.97	83.33 +/- 3.48	75.67 +/- 1.02
text-davinci-002-unknown	80.64	74.43 +/- 1.83	83.70 +/- 2.74	79.76 +/- 1.44
text-davinci-003-unknown	79.53	72.92 +/- 2.49	86.59 +/- 1.95	78.55 +/- 1.20
ChatGPT-unknown	75.64	67.99 +/- 2.74	78.26 +/- 6.15	74.55 +/- 3.90
GPT-4-unknown	82.17	71.97 +/- 2.83	86.23 +/- 3.48	82.34 +/- 2.67
Humans	86.23	83.18	92.17	84.86

Table D.29: Accuracy per label for model group Example IT for 5-shot chain-of-thought evaluation.

Model	Mean	World knowledge	Idiom	Rhetorical question
Cohere-command-6b	69.14	72.46 +/- 5.98	78.03 +/- 3.12	62.12 +/- 8.16
Cohere-command-52b	75.28	78.99 +/- 1.62	84.47 +/- 3.57	51.52 +/- 8.57
text-ada-001-unknown	15.33	11.59 +/- 8.20	17.42 +/- 9.43	10.61 +/- 9.70
text-babbage-001-unknown	47.67	47.83 +/- 11.50	55.30 +/- 16.21	42.42 +/- 8.57
text-curie-001-unknown	68.22	69.57 +/- 6.64	79.17 +/- 0.85	69.70 +/- 10.05
text-davinci-001-unknown	67.25	69.57 +/- 7.10	71.59 +/- 3.65	60.61 +/- 11.34
text-davinci-002-unknown	80.06	92.03 +/- 1.62	88.26 +/- 3.57	46.97 +/- 24.29
text-davinci-003-unknown	83.61	93.48 +/- 2.17	93.18 +/- 0.00	69.70 +/- 10.05
ChatGPT-unknown	77.19	89.86 +/- 4.10	87.88 +/- 3.63	65.15 +/- 9.70
GPT-4-unknown	86.47	93.48 +/- 3.32	93.18 +/- 2.93	87.88 +/- 4.29
Humans	86.23	93.04	92.73	92.73

Table D.30: Accuracy per label for model group Example IT for 5-shot chain-of-thought evaluation.

Model	Mean	Particularised	Generalised	Other
Cohere-command-6b	69.14	58.33 +/- 1.93	81.52 +/- 2.08	69.04 +/- 2.04
Cohere-command-52b	75.28	68.94 +/- 3.19	77.17 +/- 2.08	75.88 +/- 0.53
text-ada-001-unknown	15.33	15.53 +/- 7.73	14.86 +/- 9.26	15.50 +/- 8.33
text-babbage-001-unknown	47.67	45.27 +/- 11.94	40.94 +/- 19.34	48.19 +/- 14.11
text-curie-001-unknown	68.22	59.47 +/- 5.15	74.28 +/- 7.88	68.04 +/- 1.75
text-davinci-001-unknown	67.25	64.58 +/- 3.85	64.13 +/- 5.14	67.92 +/- 3.30
text-davinci-002-unknown	80.06	75.95 +/- 3.68	80.07 +/- 6.69	80.23 +/- 1.07
text-davinci-003-unknown	83.61	77.46 +/- 1.02	87.32 +/- 3.18	83.25 +/- 0.96
ChatGPT-unknown	77.19	72.35 +/- 1.56	80.43 +/- 5.47	76.23 +/- 1.11
GPT-4-unknown	86.47	81.63 +/- 2.58	88.77 +/- 4.05	86.05 +/- 1.17
Humans	86.23	83.18	92.17	84.86

Table D.31: Accuracy per prompt template for BERT-cased.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	47.3	48.8	50.5	49.8	46.7	46.7
2	46.8	50.3	45.5	50.2	46.7	46.5
3	57.3	51.5	50.0	50.0	47.0	46.7
4	48.8	51.0	49.5	48.5	46.8	46.7
5	46.7	50.3	44.5	47.7	46.7	46.7
6	46.7	50.3	45.8	47.8	46.8	46.7
Mean	48.9	50.4	47.6	49.0	46.8	46.7
– std	3.81	0.832	2.42	1.04	0.107	0.0745
Structured	51.1	50.4	50.0	49.4	46.8	46.7
– std	4.4	1.17	0.408	0.665	0.125	7.11e-15
Natural	46.7	50.3	45.3	48.6	46.7	46.6
– std	0.0471	7.11e-15	0.556	1.16	0.0471	0.0943

Table D.32: Accuracy per prompt template for BERT-uncased.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	57.0	53.2	51.8	55.2	51.7	49.3
2	53.7	50.3	54.0	48.7	49.0	49.3
3	54.7	54.7	57.3	55.5	53.3	52.8
4	56.7	51.5	52.3	54.0	50.3	49.5
5	53.2	50.2	50.2	48.3	48.2	47.2
6	53.3	50.3	54.2	49.2	53.0	53.5
Mean	54.8	51.7	53.3	51.8	50.9	50.3
– std	1.55	1.71	2.24	3.13	1.92	2.19
Structured	56.1	53.1	53.8	54.9	51.8	50.5
– std	1.02	1.31	2.48	0.648	1.23	1.6
Natural	53.4	50.3	52.8	48.7	50.1	50.0
– std	0.216	0.0471	1.84	0.368	2.1	2.62

Table D.33: Accuracy per prompt template for RoBERTa-base.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.0	55.8	58.0	58.7	58.3	57.8
2	56.5	50.5	52.0	55.8	56.0	54.2
3	53.0	56.8	56.8	61.3	59.5	58.8
4	55.2	56.0	58.7	59.8	56.8	57.2
5	55.7	50.3	52.3	54.8	55.5	53.0
6	59.2	50.3	54.2	55.8	55.7	55.3
Mean	55.6	53.3	55.3	57.7	57.0	56.1
– std	1.97	2.93	2.65	2.38	1.47	2.05
Structured	54.1	56.2	57.8	59.9	58.2	57.9
– std	0.899	0.432	0.785	1.07	1.1	0.66
Natural	57.1	50.4	52.8	55.5	55.7	54.2
– std	1.5	0.0943	0.974	0.471	0.205	0.939

Table D.34: Accuracy per prompt template for RoBERTa-large.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	57.7	50.2	62.0	64.7	64.7	60.5
2	46.7	53.3	58.5	64.2	61.2	55.7
3	60.8	54.8	64.5	62.8	61.8	59.5
4	66.2	50.3	64.0	59.0	57.0	58.2
5	46.7	53.3	58.8	63.5	60.5	56.5
6	46.7	55.5	59.3	60.0	60.8	52.3
Mean	54.1	52.9	61.2	62.4	61.0	57.1
– std	7.84	2.03	2.45	2.13	2.26	2.7
Structured	61.6	51.8	63.5	62.2	61.2	59.4
– std	3.51	2.15	1.08	2.37	3.18	0.942
Natural	46.7	54.0	58.9	62.6	60.8	54.8
– std	7.11e-15	1.04	0.33	1.84	0.287	1.82

Table D.35: Accuracy per prompt template for GPT-2-medium.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.2	53.7	54.0	53.8	53.8	55.0
2	52.8	53.7	55.8	57.2	60.3	57.2
3	53.7	54.0	52.5	56.5	55.8	55.3
4	53.5	55.7	53.3	55.8	55.5	54.3
5	59.2	54.3	56.7	57.7	60.7	58.8
6	58.3	54.8	55.7	57.7	61.7	57.8
Mean	55.1	54.4	54.7	56.4	58.0	56.4
– std	2.6	0.706	1.5	1.36	3.03	1.63
Structured	53.5	54.5	53.3	55.4	55.0	54.9
– std	0.205	0.881	0.613	1.14	0.881	0.419
Natural	56.8	54.3	56.1	57.5	60.9	57.9
– std	2.83	0.45	0.45	0.236	0.589	0.66

Table D.36: Accuracy per prompt template for GPT-2-large.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.3	53.3	54.5	53.5	55.3	56.2
2	47.5	56.7	57.5	57.8	60.8	61.0
3	55.0	53.8	55.7	54.0	54.8	56.0
4	54.0	53.7	56.2	53.5	54.8	56.7
5	47.2	54.5	56.7	58.8	61.2	60.8
6	47.0	53.3	57.2	59.5	60.3	60.8
Mean	50.7	54.2	56.3	56.2	57.9	58.6
– std	3.47	1.18	1.0	2.57	2.92	2.29
Structured	54.1	53.6	55.5	53.7	55.0	56.3
– std	0.698	0.216	0.713	0.236	0.236	0.294
Natural	47.2	54.8	57.1	58.7	60.8	60.9
– std	0.205	1.41	0.33	0.698	0.368	0.0943

Table D.37: Accuracy per prompt template for GPT-2-xl.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.2	53.3	57.0	54.5	54.7	56.2
2	48.7	61.3	57.3	63.7	62.0	60.5
3	55.0	55.2	59.5	59.0	58.0	60.7
4	54.2	54.3	56.0	54.5	54.3	56.3
5	48.0	59.7	58.3	60.8	62.7	61.7
6	48.5	60.8	58.0	61.8	61.5	61.5
Mean	51.3	57.4	57.7	59.1	58.9	59.5
– std	2.92	3.25	1.1	3.5	3.43	2.32
Structured	54.1	54.3	57.5	56.0	55.7	57.7
– std	0.736	0.776	1.47	2.12	1.66	2.1
Natural	48.4	60.6	57.9	62.1	62.1	61.2
– std	0.294	0.668	0.419	1.2	0.492	0.525

Table D.38: Accuracy per prompt template for EleutherAI-125M.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.3	53.7	52.7	56.2	56.2	54.0
2	52.2	50.0	47.5	53.5	55.7	53.3
3	53.3	53.8	51.2	55.8	54.8	52.8
4	53.7	52.5	51.2	53.8	55.8	53.2
5	50.7	50.2	47.3	53.8	56.2	53.8
6	48.2	49.8	47.5	53.2	57.5	53.5
Mean	51.9	51.7	49.6	54.4	56.0	53.4
– std	1.93	1.72	2.19	1.17	0.806	0.394
Structured	53.4	53.3	51.7	55.3	55.6	53.3
– std	0.189	0.591	0.707	1.05	0.589	0.499
Natural	50.4	50.0	47.4	53.5	56.5	53.5
– std	1.65	0.163	0.0943	0.245	0.759	0.205

Table D.39: Accuracy per prompt template for EleutherAI-1.3B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.3	53.7	54.8	57.5	57.2	56.2
2	51.8	56.8	57.5	59.0	55.8	54.7
3	58.0	55.5	59.5	58.0	61.5	57.5
4	53.2	57.5	56.8	55.2	56.5	54.7
5	49.7	55.2	57.5	58.7	57.2	56.7
6	51.8	55.7	56.5	58.7	56.5	56.2
Mean	53.1	55.7	57.1	57.8	57.4	56.0
– std	2.59	1.21	1.4	1.29	1.87	1.02
Structured	55.2	55.6	57.0	56.9	58.4	56.1
– std	2.05	1.55	1.93	1.22	2.21	1.14
Natural	51.1	55.9	57.2	58.8	56.5	55.9
– std	0.99	0.668	0.471	0.141	0.572	0.85

Table D.40: Accuracy per prompt template for EleutherAI-2.7B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.0	52.8	58.2	57.8	59.5	56.7
2	62.0	56.2	57.7	55.8	57.8	57.7
3	58.7	60.0	58.8	59.2	57.8	57.8
4	56.5	54.2	57.5	56.2	57.5	55.5
5	62.7	54.7	58.7	55.7	57.3	57.8
6	61.2	55.2	57.3	57.5	58.5	58.7
Mean	59.2	55.5	58.0	57.0	58.1	57.4
– std	3.13	2.25	0.576	1.26	0.741	1.02
Structured	56.4	55.7	58.2	57.7	58.3	56.7
– std	1.92	3.12	0.531	1.23	0.881	0.939
Natural	62.0	55.4	57.9	56.3	57.9	58.1
– std	0.613	0.624	0.589	0.826	0.492	0.45

Table D.41: Accuracy per prompt template for EleutherAI-6B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	57.5	58.8	52.7	53.0	52.5	51.3
2	57.7	51.8	63.2	62.7	64.3	65.3
3	56.2	58.2	57.2	53.0	54.7	54.5
4	52.8	55.5	53.3	52.2	54.0	53.8
5	56.8	52.7	62.7	63.2	65.2	64.2
6	57.2	52.8	61.3	61.8	62.2	63.3
Mean	56.4	55.0	58.4	57.6	58.8	58.7
– std	1.67	2.75	4.28	4.94	5.2	5.65
Structured	55.5	57.5	54.4	52.7	53.7	53.2
– std	1.98	1.44	1.99	0.377	0.918	1.37
Natural	57.2	52.4	62.4	62.6	63.9	64.3
– std	0.368	0.45	0.804	0.579	1.26	0.818

Table D.42: Accuracy per prompt template for EleutherAI-20B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.0	58.0	55.3	54.3	52.8	54.3
2	61.3	54.2	65.8	63.3	65.0	60.3
3	54.3	58.3	58.5	56.7	55.3	52.0
4	56.2	58.2	55.3	57.2	57.0	58.7
5	59.0	53.0	66.7	62.8	65.0	59.2
6	61.3	53.5	65.2	61.7	64.0	59.7
Mean	57.5	55.9	61.1	59.3	59.9	57.4
– std	3.25	2.33	4.9	3.42	4.98	3.09
Structured	54.5	58.2	56.4	56.1	55.0	55.0
– std	1.31	0.125	1.51	1.27	1.72	2.78
Natural	60.5	53.6	65.9	62.6	64.7	59.7
– std	1.08	0.492	0.616	0.668	0.471	0.45

Table D.43: Accuracy per prompt template for BLOOM-560M.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.3	54.2	53.5	53.8	53.8	53.5
2	46.7	56.3	54.0	54.8	56.0	55.3
3	58.8	53.3	53.8	53.3	54.5	54.0
4	56.3	54.8	53.5	54.8	52.7	56.7
5	46.7	54.3	53.7	55.3	56.3	55.5
6	46.7	56.0	54.0	55.2	56.7	55.0
Mean	51.6	54.8	53.8	54.5	55.0	55.0
– std	5.05	1.04	0.206	0.734	1.45	1.04
Structured	56.5	54.1	53.6	54.0	53.7	54.7
– std	1.84	0.616	0.141	0.624	0.741	1.41
Natural	46.7	55.5	53.9	55.1	56.3	55.3
– std	7.11e-15	0.881	0.141	0.216	0.287	0.205

Table D.44: Accuracy per prompt template for BLOOM-1B1.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.3	53.5	56.2	54.2	55.2	54.5
2	49.0	51.5	58.2	59.8	58.8	60.8
3	57.2	54.2	55.8	54.0	55.5	50.8
4	53.3	54.0	54.2	53.3	55.7	55.8
5	47.3	51.2	59.8	61.3	60.2	60.0
6	46.8	51.0	60.2	61.2	60.2	59.3
Mean	51.2	52.6	57.4	57.3	57.6	56.9
– std	3.75	1.36	2.18	3.51	2.19	3.53
Structured	54.6	53.9	55.4	53.8	55.5	53.7
– std	1.84	0.294	0.864	0.386	0.205	2.12
Natural	47.7	51.2	59.4	60.8	59.7	60.0
– std	0.942	0.205	0.864	0.685	0.66	0.613

Table D.45: Accuracy per prompt template for BLOOM-1B7.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.5	54.7	53.8	54.0	55.7	56.5
2	57.7	52.2	56.3	55.5	55.8	52.0
3	54.7	53.2	53.8	51.0	54.5	54.0
4	54.5	53.8	54.5	51.2	55.5	50.3
5	50.0	51.2	54.3	53.2	54.7	50.0
6	51.3	51.8	53.8	54.0	54.7	50.8
Mean	53.6	52.8	54.4	53.1	55.1	52.3
– std	2.49	1.2	0.886	1.6	0.528	2.31
Structured	54.2	53.9	54.0	52.1	55.2	53.6
– std	0.525	0.616	0.33	1.37	0.525	2.55
Natural	53.0	51.7	54.8	54.2	55.1	50.9
– std	3.37	0.411	1.08	0.953	0.519	0.822

Table D.46: Accuracy per prompt template for BLOOM-3B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.0	54.0	56.8	59.5	60.0	58.2
2	62.5	58.0	58.2	59.7	57.5	60.0
3	53.5	54.0	57.2	58.7	59.2	58.2
4	54.8	55.3	55.7	59.0	58.2	55.8
5	58.5	57.5	58.0	59.7	58.8	60.2
6	59.0	56.8	57.3	59.8	58.5	59.5
Mean	56.9	55.9	57.2	59.4	58.7	58.6
– std	3.4	1.6	0.823	0.408	0.783	1.5
Structured	53.8	54.4	56.6	59.1	59.1	57.4
– std	0.759	0.613	0.634	0.33	0.736	1.13
Natural	60.0	57.4	57.8	59.7	58.3	59.9
– std	1.78	0.492	0.386	0.0471	0.556	0.294

Table D.47: Accuracy per prompt template for BLOOM-7B1.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.2	55.2	55.2	52.0	53.0	52.7
2	61.2	59.0	53.7	58.3	58.8	61.7
3	58.7	53.3	53.0	53.3	53.0	52.8
4	53.5	53.5	55.2	52.8	54.3	53.5
5	62.0	61.0	55.3	60.3	58.5	62.5
6	63.5	60.0	54.7	59.8	56.3	62.5
Mean	58.7	57.0	54.5	56.1	55.7	57.6
– std	4.03	3.11	0.871	3.46	2.39	4.63
Structured	55.1	54.0	54.5	52.7	53.4	53.0
– std	2.52	0.852	1.04	0.535	0.613	0.356
Natural	62.2	60.0	54.6	59.5	57.9	62.2
– std	0.953	0.816	0.66	0.85	1.11	0.377

Table D.48: Accuracy per prompt template for BLOOM-176B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.8	58.8	58.5	57.7	55.7	56.7
2	55.8	60.8	68.0	65.7	64.2	62.7
3	53.5	66.7	69.3	71.8	71.7	69.8
4	54.3	59.8	64.8	62.2	60.7	61.3
5	52.3	61.3	66.2	61.8	58.8	57.5
6	55.5	59.2	65.7	61.7	60.3	58.3
Mean	54.2	61.1	65.4	63.5	61.9	61.1
– std	1.19	2.65	3.43	4.38	5.06	4.44
Structured	53.9	61.8	64.2	63.9	62.7	62.6
– std	0.33	3.51	4.43	5.88	6.68	5.43
Natural	54.5	60.4	66.6	63.1	61.1	59.5
– std	1.58	0.896	0.988	1.86	2.28	2.29

Table D.49: Accuracy per prompt template for OPT-125M.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.3	55.2	54.0	55.2	54.2	55.0
2	49.5	50.5	47.5	52.7	50.5	48.2
3	53.5	55.5	53.0	55.0	53.7	56.0
4	53.3	54.5	54.2	53.8	54.3	53.8
5	48.5	50.5	46.3	50.7	49.5	48.0
6	47.3	50.2	46.3	50.0	49.0	48.0
Mean	50.9	52.7	50.2	52.9	51.9	51.5
– std	2.55	2.35	3.56	1.99	2.25	3.49
Structured	53.4	55.1	53.7	54.7	54.1	54.9
– std	0.0943	0.419	0.525	0.618	0.262	0.899
Natural	48.4	50.4	46.7	51.1	49.7	48.1
– std	0.899	0.141	0.566	1.14	0.624	0.0943

Table D.50: Accuracy per prompt template for OPT-350M.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.3	53.8	51.5	56.5	54.2	54.7
2	60.5	50.3	50.8	56.5	55.2	54.0
3	53.3	56.3	52.8	58.7	55.0	56.2
4	53.7	56.3	52.0	55.2	55.2	56.3
5	62.3	50.3	50.8	57.0	56.5	53.5
6	59.7	50.3	50.8	56.5	56.5	53.0
Mean	57.1	52.9	51.4	56.7	55.4	54.6
– std	3.78	2.71	0.752	1.04	0.826	1.26
Structured	53.4	55.5	52.1	56.8	54.8	55.7
– std	0.189	1.18	0.535	1.44	0.432	0.732
Natural	60.8	50.3	50.8	56.7	56.1	53.5
– std	1.09	7.11e-15	7.11e-15	0.236	0.613	0.408

Table D.51: Accuracy per prompt template for OPT-1.3B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	57.8	56.2	55.5	60.2	59.8	62.7
2	62.2	57.0	61.2	61.8	64.8	67.2
3	60.8	59.5	57.2	59.7	60.3	58.2
4	54.8	55.8	59.2	56.5	57.0	54.7
5	62.5	56.2	59.3	61.7	65.0	64.5
6	64.0	53.2	55.8	59.7	62.7	62.8
Mean	60.4	56.3	58.0	59.9	61.6	61.7
– std	3.13	1.85	2.05	1.76	2.86	4.11
Structured	57.8	57.2	57.3	58.8	59.0	58.5
– std	2.45	1.66	1.51	1.64	1.45	3.27
Natural	62.9	55.5	58.8	61.1	64.2	64.8
– std	0.787	1.64	2.24	0.967	1.04	1.81

Table D.52: Accuracy per prompt template for OPT-2.7B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.7	53.0	53.2	53.8	54.3	53.7
2	64.0	60.3	60.2	60.3	61.3	64.5
3	55.8	53.3	55.2	55.8	57.0	56.5
4	54.5	53.3	54.8	55.5	56.8	57.0
5	64.8	60.7	60.7	62.2	64.3	64.3
6	63.5	60.3	60.0	60.5	63.3	63.2
Mean	59.6	56.8	57.4	58.0	59.5	59.9
– std	4.58	3.62	3.02	3.11	3.68	4.28
Structured	55.0	53.2	54.4	55.0	56.0	55.7
– std	0.572	0.141	0.864	0.881	1.23	1.45
Natural	64.1	60.4	60.3	61.0	63.0	64.0
– std	0.535	0.189	0.294	0.852	1.25	0.572

Table D.53: Accuracy per prompt template for OPT-6.7B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	55.7	54.3	60.8	61.2	61.2	58.5
2	64.2	68.0	66.8	65.7	66.3	66.3
3	54.2	53.5	59.5	61.2	63.3	60.5
4	58.8	56.3	61.8	62.2	63.5	63.2
5	64.2	65.2	66.0	65.2	67.7	67.5
6	65.0	63.2	64.8	64.3	66.3	65.7
Mean	60.4	60.1	63.3	63.3	64.7	63.6
– std	4.34	5.62	2.73	1.84	2.23	3.23
Structured	56.2	54.7	60.7	61.5	62.7	60.7
– std	1.92	1.18	0.942	0.471	1.04	1.93
Natural	64.5	65.5	65.9	65.1	66.8	66.5
– std	0.377	1.97	0.822	0.579	0.66	0.748

Table D.54: Accuracy per prompt template for OPT-13B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.7	64.0	69.8	68.2	67.8	62.2
2	68.2	57.8	69.5	68.0	66.8	63.7
3	54.3	62.2	65.2	63.2	64.3	66.3
4	58.3	63.3	64.3	63.7	63.5	64.0
5	66.0	58.5	67.2	65.3	63.7	62.7
6	64.7	57.5	68.3	66.2	64.8	61.5
Mean	61.0	60.6	67.4	65.8	65.1	63.4
– std	5.51	2.68	2.06	1.92	1.6	1.55
Structured	55.8	63.2	66.4	65.0	65.2	64.2
– std	1.8	0.741	2.41	2.25	1.87	1.68
Natural	66.3	57.9	68.3	66.5	65.1	62.6
– std	1.44	0.419	0.939	1.12	1.28	0.899

Table D.55: Accuracy per prompt template for OPT-30B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	62.2	62.7	66.0	65.2	65.5	65.0
2	62.0	58.7	69.0	65.7	66.3	69.0
3	60.3	63.5	62.7	60.8	60.5	61.5
4	65.0	66.8	57.8	57.2	57.2	56.2
5	60.3	55.8	70.0	66.0	67.2	71.0
6	59.0	54.5	68.3	65.3	67.7	70.2
Mean	61.5	60.3	65.6	63.4	64.1	65.5
– std	1.92	4.37	4.24	3.27	3.87	5.28
Structured	62.5	64.3	62.2	61.1	61.1	60.9
– std	1.93	1.77	3.37	3.27	3.41	3.62
Natural	60.4	56.3	69.1	65.7	67.1	70.1
– std	1.23	1.76	0.698	0.287	0.579	0.822

Table D.56: Accuracy per prompt template for OPT-66B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	59.3	56.2	56.7	56.5	55.7	54.3
2	66.5	67.3	65.3	64.2	67.2	65.2
3	56.5	64.3	55.5	55.0	56.2	52.2
4	62.0	61.5	66.5	63.0	61.7	63.7
5	62.5	66.0	64.8	63.7	65.7	65.0
6	61.2	63.8	60.2	62.5	64.7	64.7
Mean	61.3	63.2	61.5	60.8	61.9	60.8
– std	3.06	3.61	4.3	3.65	4.5	5.43
Structured	59.3	60.7	59.6	58.2	57.9	56.7
– std	2.25	3.36	4.93	3.47	2.72	5.0
Natural	63.4	65.7	63.4	63.5	65.9	65.0
– std	2.26	1.44	2.3	0.713	1.03	0.205

Table D.57: Accuracy per prompt template for OPT-175B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	56.7	58.0	64.8	61.0	65.0	62.3
2	52.7	53.3	67.3	63.2	68.0	65.8
3	54.5	68.5	60.0	55.3	57.8	56.7
4	64.0	66.7	61.5	58.0	62.0	58.7
5	52.0	52.0	65.0	63.8	67.8	65.2
6	52.2	51.7	64.7	63.2	68.0	66.0
Mean	55.3	58.4	63.9	60.8	64.8	62.4
– std	4.19	6.87	2.42	3.13	3.79	3.62
Structured	58.4	64.4	62.1	58.1	61.6	59.2
– std	4.06	4.58	2.0	2.33	2.95	2.32
Natural	52.3	52.3	65.7	63.4	67.9	65.7
– std	0.294	0.694	1.16	0.283	0.0943	0.34

Table D.58: Accuracy per prompt template for Cohere-409.3M (Cohere-small).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.2	49.7	52.7	51.7	53.5	56.0
2	47.5	50.7	52.7	53.2	55.8	57.8
3	57.2	55.5	55.2	55.5	55.7	57.0
4	54.8	53.8	54.5	56.8	54.8	54.5
5	48.5	50.7	52.8	52.7	56.0	58.8
6	47.5	51.0	52.5	53.7	55.3	58.8
Mean	51.6	51.9	53.4	53.9	55.2	57.2
– std	3.91	2.05	1.05	1.72	0.847	1.54
Structured	55.4	53.0	54.1	54.7	54.7	55.8
– std	1.3	2.43	1.05	2.16	0.903	1.03
Natural	47.8	50.8	52.7	53.2	55.7	58.5
– std	0.471	0.141	0.125	0.408	0.294	0.471

Table D.59: Accuracy per prompt template for Cohere-6.067B (Cohere-medium).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.7	54.2	55.3	51.8	56.3	55.3
2	61.8	62.8	64.3	63.8	65.2	64.7
3	57.2	53.3	58.5	55.3	57.8	55.3
4	56.0	53.3	57.0	53.2	55.8	56.7
5	57.8	60.7	64.0	64.2	64.7	64.2
6	56.2	62.8	66.2	64.0	62.8	66.0
Mean	57.3	57.9	60.9	58.7	60.4	60.4
– std	2.24	4.32	4.11	5.38	3.92	4.65
Structured	56.0	53.6	56.9	53.4	56.6	55.8
– std	1.02	0.424	1.31	1.44	0.85	0.66
Natural	58.6	62.1	64.8	64.0	64.2	65.0
– std	2.36	0.99	0.974	0.163	1.03	0.759

Table D.60: Accuracy per prompt template for Cohere-13.12B (Cohere-large).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	55.3	57.3	56.3	55.0	58.5	59.0
2	59.2	64.2	68.0	66.3	64.7	69.5
3	57.2	62.8	61.0	59.0	64.2	62.3
4	55.5	61.3	56.3	54.0	59.0	59.8
5	56.8	64.3	66.7	64.2	65.7	69.8
6	59.2	60.7	66.5	63.7	65.0	68.3
Mean	57.2	61.8	62.5	60.4	62.9	64.8
– std	1.56	2.41	4.88	4.69	2.94	4.55
Structured	56.0	60.5	57.9	56.0	60.6	60.4
– std	0.852	2.32	2.22	2.16	2.58	1.41
Natural	58.4	63.1	67.1	64.7	65.1	69.2
– std	1.13	1.67	0.665	1.13	0.419	0.648

Table D.61: Accuracy per prompt template for Cohere-52B (Cohere-xl).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	56.0	60.7	70.3	65.3	66.3	68.7
2	62.8	65.0	64.3	64.2	65.0	64.3
3	54.0	65.3	62.8	60.2	64.0	63.5
4	53.8	55.5	61.8	64.8	64.3	64.7
5	62.2	65.7	67.3	63.0	63.7	65.3
6	62.2	65.7	64.2	62.3	65.0	67.8
Mean	58.5	63.0	65.1	63.3	64.7	65.7
– std	3.97	3.77	2.87	1.72	0.855	1.89
Structured	54.6	60.5	65.0	63.4	64.9	65.6
– std	0.993	4.0	3.79	2.3	1.02	2.22
Natural	62.4	65.5	65.3	63.2	64.6	65.8
– std	0.283	0.33	1.44	0.785	0.613	1.47

Table D.62: Accuracy per prompt template for GPT-3-350M (ada).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	55.3	57.2	58.3	57.5	58.2	60.5
2	46.7	56.8	56.3	59.5	59.2	61.7
3	54.0	54.5	53.3	54.0	56.5	56.7
4	53.5	52.8	54.7	56.7	58.8	59.7
5	49.8	57.3	55.3	58.5	58.8	61.8
6	49.5	57.2	56.3	60.2	61.5	61.2
Mean	51.5	56.0	55.7	57.7	58.8	60.3
– std	3.02	1.72	1.55	2.04	1.48	1.75
Structured	54.3	54.8	55.4	56.1	57.8	59.0
– std	0.759	1.81	2.11	1.5	0.974	1.64
Natural	48.7	57.1	56.0	59.4	59.8	61.6
– std	1.4	0.216	0.471	0.698	1.19	0.262

Table D.63: Accuracy per prompt template for GPT-3-1.3B (babbage).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	55.7	60.7	61.0	59.0	60.7	57.8
2	63.0	62.5	65.7	61.7	63.0	59.3
3	56.2	59.0	60.5	59.3	64.8	61.0
4	53.3	59.7	60.7	62.5	65.0	66.7
5	59.2	62.5	63.7	61.8	61.5	58.7
6	59.0	60.2	64.3	61.2	62.2	57.7
Mean	57.7	60.8	62.6	60.9	62.9	60.2
– std	3.1	1.33	2.01	1.31	1.6	3.11
Structured	55.1	59.8	60.7	60.3	63.5	61.8
– std	1.27	0.698	0.205	1.58	1.98	3.68
Natural	60.4	61.7	64.6	61.6	62.2	58.6
– std	1.84	1.08	0.838	0.262	0.613	0.66

Table D.64: Accuracy per prompt template for GPT-3-6.7B (curie).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.3	58.3	63.0	64.8	67.7	64.0
2	57.5	65.2	63.2	65.3	65.8	65.2
3	57.0	54.2	59.2	61.2	60.8	59.3
4	53.3	61.7	62.8	63.8	64.7	60.7
5	55.3	64.2	62.5	64.5	65.8	63.7
6	52.5	63.5	63.7	64.0	66.2	64.3
Mean	54.8	61.2	62.4	63.9	65.2	62.9
– std	1.92	3.83	1.48	1.32	2.14	2.12
Structured	54.5	58.1	61.7	63.3	64.4	61.3
– std	1.74	3.07	1.75	1.52	2.82	1.97
Natural	55.1	64.3	63.1	64.6	65.9	64.4
– std	2.05	0.698	0.492	0.535	0.189	0.616

Table D.65: Accuracy per prompt template for GPT-3-175B (davinci).

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	61.2	67.3	66.3	62.7	66.7	66.2
2	53.7	65.3	68.8	69.3	71.0	69.7
3	58.7	65.8	68.2	64.7	65.0	65.3
4	64.0	62.8	71.3	68.7	66.2	67.8
5	54.2	66.3	69.0	70.0	70.0	70.8
6	51.7	66.7	68.7	68.3	71.0	70.0
Mean	57.2	65.7	68.7	67.3	68.3	68.3
– std	4.4	1.44	1.46	2.65	2.43	2.03
Structured	61.3	65.3	68.6	65.4	66.0	66.4
– std	2.16	1.87	2.06	2.49	0.713	1.03
Natural	53.2	66.1	68.8	69.2	70.7	70.2
– std	1.08	0.589	0.125	0.698	0.471	0.464

Table D.66: Accuracy per prompt template for BlenderBot-90M.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	46.7	51.5	46.7	46.7	46.5	46.5
2	46.7	51.3	46.5	46.7	46.7	46.7
3	46.7	46.7	46.7	46.7	46.3	46.8
4	46.7	46.7	46.7	46.7	46.5	46.7
5	46.7	50.0	46.7	46.7	46.7	46.7
6	46.5	53.5	46.3	46.7	46.7	46.7
Mean	46.7	49.9	46.6	46.7	46.6	46.7
– std	0.0745	2.52	0.153	7.11e-15	0.149	0.0898
Structured	46.7	48.3	46.7	46.7	46.4	46.7
– std	7.11e-15	2.26	7.11e-15	7.11e-15	0.0943	0.125
Natural	46.6	51.6	46.5	46.7	46.7	46.7
– std	0.0943	1.44	0.163	7.11e-15	7.11e-15	7.11e-15

Table D.67: Accuracy per prompt template for BlenderBot-2.7B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.0	53.2	53.3	53.0	52.8	53.3
2	53.3	53.3	53.3	53.3	53.3	53.3
3	53.2	53.2	53.3	53.2	53.2	53.2
4	53.5	53.5	53.5	53.3	52.8	53.0
5	53.3	53.3	53.3	53.3	53.3	53.3
6	53.3	53.3	53.3	53.3	53.3	53.3
Mean	53.4	53.3	53.3	53.2	53.1	53.2
– std	0.269	0.1	0.0745	0.111	0.227	0.111
Structured	53.6	53.3	53.4	53.2	52.9	53.2
– std	0.33	0.141	0.0943	0.125	0.189	0.125
Natural	53.3	53.3	53.3	53.3	53.3	53.3
– std	7.11e-15	7.11e-15	7.11e-15	7.11e-15	7.11e-15	7.11e-15

Table D.68: Accuracy per prompt template for BlenderBot-9.4B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	53.7	51.5	53.0	53.0	53.0	54.0
2	53.2	53.8	54.2	52.5	52.2	52.2
3	53.3	49.7	52.0	54.0	54.2	55.5
4	54.0	55.3	52.5	54.0	53.5	53.7
5	53.3	52.8	53.5	53.2	53.5	53.3
6	52.7	52.0	51.7	53.5	52.8	53.7
Mean	53.4	52.5	52.8	53.4	53.2	53.7
– std	0.407	1.77	0.859	0.537	0.63	0.978
Structured	53.7	52.2	52.5	53.7	53.6	54.4
– std	0.287	2.33	0.408	0.471	0.492	0.787
Natural	53.1	52.9	53.1	53.1	52.8	53.1
– std	0.262	0.736	1.05	0.419	0.531	0.634

Table D.69: Accuracy per prompt template for T0-3B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	48.7	49.5	46.5	46.7	46.7	46.7
2	46.7	47.5	46.7	46.7	46.7	46.7
3	49.2	48.3	46.7	46.7	46.7	46.7
4	51.7	49.0	46.7	46.7	46.7	46.7
5	46.7	49.2	46.7	46.7	46.7	46.7
6	46.7	49.8	46.8	46.7	46.7	46.7
Mean	48.3	48.9	46.7	46.7	46.7	46.7
– std	1.84	0.773	0.0898	7.11e-15	7.11e-15	7.11e-15
Structured	49.9	48.9	46.6	46.7	46.7	46.7
– std	1.31	0.492	0.0943	7.11e-15	7.11e-15	7.11e-15
Natural	46.7	48.8	46.7	46.7	46.7	46.7
– std	7.11e-15	0.974	0.0471	7.11e-15	7.11e-15	7.11e-15

Table D.70: Accuracy per prompt template for T0-11B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	57.5	47.7	47.3	46.8	46.7	46.7
2	49.3	47.5	46.7	46.7	46.8	46.7
3	65.3	48.8	47.3	46.7	46.7	46.7
4	63.8	48.0	47.0	46.7	46.7	46.7
5	48.0	47.2	46.7	46.7	47.0	46.8
6	49.7	47.5	47.0	46.8	47.0	47.0
Mean	55.6	47.8	47.0	46.7	46.8	46.8
– std	7.04	0.515	0.245	0.0471	0.134	0.111
Structured	62.2	48.2	47.2	46.7	46.7	46.7
– std	3.38	0.464	0.141	0.0471	7.11e-15	7.11e-15
Natural	49.0	47.4	46.8	46.7	46.9	46.8
– std	0.726	0.141	0.141	0.0471	0.0943	0.125

Table D.71: Accuracy per prompt template for Flan-T5-780M.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	64.5	63.3	62.2	60.7	61.5	60.2
2	66.5	65.8	65.3	62.8	65.5	65.0
3	61.7	60.2	58.8	60.8	59.8	59.7
4	58.0	50.2	50.7	51.3	52.3	54.8
5	63.8	69.0	64.3	63.2	65.2	65.5
6	65.3	68.8	64.8	62.3	64.7	63.8
Mean	63.3	62.9	61.0	60.2	61.5	61.5
– std	2.79	6.44	5.1	4.08	4.61	3.73
Structured	61.4	57.9	57.2	57.6	57.9	58.2
– std	2.66	5.59	4.82	4.45	4.0	2.44
Natural	65.2	67.9	64.8	62.8	65.1	64.8
– std	1.1	1.46	0.408	0.368	0.33	0.713

Table D.72: Accuracy per prompt template for Flan-T5-3B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	54.7	58.8	56.8	56.7	57.5	60.0
2	51.2	50.8	59.0	59.2	59.0	59.7
3	54.8	51.3	49.7	49.0	48.7	48.5
4	55.3	50.0	48.0	49.0	49.3	50.8
5	51.0	54.3	57.2	58.0	58.0	57.8
6	48.0	51.2	58.7	59.0	58.0	59.8
Mean	52.5	52.7	54.9	55.1	55.1	56.1
– std	2.65	3.02	4.37	4.42	4.33	4.67
Structured	54.9	53.4	51.5	51.6	51.8	53.1
– std	0.262	3.88	3.81	3.63	4.01	4.97
Natural	50.1	52.1	58.3	58.7	58.3	59.1
– std	1.46	1.56	0.787	0.525	0.471	0.92

Table D.73: Accuracy per prompt template for Flan-T5-11B.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	64.3	61.0	63.7	65.0	62.5	64.3
2	61.5	59.7	63.2	62.3	64.0	68.0
3	56.5	63.0	60.2	57.3	56.7	56.8
4	61.7	47.7	51.7	50.3	50.3	49.5
5	61.5	55.8	64.8	64.7	65.5	66.3
6	59.2	57.5	66.3	63.7	66.0	67.7
Mean	60.8	57.4	61.7	60.5	60.8	62.1
– std	2.42	4.94	4.82	5.25	5.62	6.78
Structured	60.8	57.2	58.5	57.5	56.5	56.9
– std	3.24	6.79	5.04	6.0	4.98	6.04
Natural	60.7	57.7	64.8	63.6	65.2	67.3
– std	1.08	1.6	1.27	0.984	0.85	0.741

Table D.74: Accuracy per prompt template for Cohere-command-6b.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	65.0	63.2	71.7	70.2	71.3	70.3
2	64.8	64.2	66.8	67.5	69.8	71.7
3	68.0	65.5	69.2	65.5	66.8	68.2
4	70.0	68.5	69.2	71.2	71.7	73.2
5	66.3	65.0	66.8	67.5	70.5	69.8
6	63.7	63.7	67.7	67.5	70.0	69.5
Mean	66.3	65.0	68.6	68.2	70.0	70.5
– std	2.13	1.74	1.71	1.9	1.59	1.61
Structured	67.7	65.7	70.0	69.0	69.9	70.6
– std	2.05	2.17	1.18	2.49	2.22	2.05
Natural	64.9	64.3	67.1	67.5	70.1	70.3
– std	1.07	0.535	0.424	0.0	0.294	0.974

Table D.75: Accuracy per prompt template for Cohere-command-52b.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	65.2	74.2	77.8	75.8	75.5	76.0
2	61.7	72.0	73.5	75.3	75.2	74.5
3	56.7	74.5	77.3	77.2	76.5	75.0
4	68.2	70.7	76.0	74.8	75.3	75.3
5	54.8	72.7	74.8	76.2	74.8	74.7
6	54.8	73.0	73.0	74.5	75.0	74.5
Mean	60.2	72.8	75.4	75.6	75.4	75.0
– std	5.19	1.29	1.8	0.903	0.546	0.529
Structured	63.4	73.1	77.0	75.9	75.8	75.4
– std	4.87	1.72	0.759	0.984	0.525	0.419
Natural	57.1	72.6	73.8	75.3	75.0	74.6
– std	3.25	0.419	0.759	0.694	0.163	0.0943

Table D.76: Accuracy per prompt template for text-ada-001-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	60.8	62.8	60.8	59.0	58.7	58.8
2	50.7	56.3	54.8	56.0	57.7	52.7
3	63.7	58.5	60.8	59.0	56.7	57.5
4	61.8	56.3	59.3	58.3	61.0	56.7
5	53.3	55.5	55.2	55.7	58.0	54.3
6	48.7	54.7	54.7	56.2	57.7	53.5
Mean	56.5	57.3	57.6	57.4	58.3	55.6
– std	5.82	2.7	2.75	1.43	1.34	2.22
Structured	62.1	59.2	60.3	58.8	58.8	57.7
– std	1.2	2.7	0.707	0.33	1.76	0.865
Natural	50.9	55.5	54.9	56.0	57.8	53.5
– std	1.88	0.653	0.216	0.205	0.141	0.653

Table D.77: Accuracy per prompt template for text-babbage-001-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	67.5	64.0	66.3	63.0	64.0	64.7
2	63.0	62.5	66.2	64.2	66.5	68.2
3	65.3	65.2	66.0	63.2	64.7	64.5
4	65.2	63.5	65.7	62.7	63.0	64.8
5	61.8	64.3	66.5	64.0	66.3	67.8
6	64.0	63.8	66.2	64.2	66.7	66.0
Mean	64.5	63.9	66.1	63.6	65.2	66.0
– std	1.82	0.815	0.25	0.605	1.4	1.5
Structured	66.0	64.2	66.0	63.0	63.9	64.7
– std	1.06	0.713	0.245	0.205	0.698	0.125
Natural	62.9	63.5	66.3	64.1	66.5	67.3
– std	0.899	0.759	0.141	0.0943	0.163	0.957

Table D.78: Accuracy per prompt template for text-curie-001-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	70.7	70.2	72.5	70.8	70.8	70.7
2	66.5	59.3	70.3	69.7	68.3	71.2
3	73.2	70.2	73.5	69.7	71.8	69.7
4	71.3	68.0	71.0	69.8	71.0	69.0
5	65.5	58.8	70.0	70.2	68.5	70.7
6	66.5	59.8	70.7	70.8	69.0	70.8
Mean	69.0	64.4	71.3	70.2	69.9	70.4
– std	2.9	5.14	1.25	0.478	1.35	0.754
Structured	71.7	69.5	72.3	70.1	71.2	69.8
– std	1.07	1.04	1.03	0.497	0.432	0.698
Natural	66.2	59.3	70.3	70.2	68.6	70.9
– std	0.471	0.408	0.287	0.45	0.294	0.216

Table D.79: Accuracy per prompt template for text-davinci-001-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	76.5	73.7	75.7	75.7	76.3	76.8
2	72.0	72.5	74.3	75.2	76.0	75.3
3	74.8	74.2	75.7	77.2	75.8	76.8
4	68.0	70.2	72.8	72.8	73.3	75.0
5	72.5	73.2	74.3	74.3	75.3	75.7
6	70.0	72.7	74.3	74.7	75.0	75.3
Mean	72.3	72.7	74.5	75.0	75.3	75.8
– std	2.82	1.28	0.991	1.34	0.986	0.724
Structured	73.1	72.7	74.7	75.2	75.1	76.2
– std	3.67	1.78	1.37	1.83	1.31	0.849
Natural	71.5	72.8	74.3	74.7	75.4	75.4
– std	1.08	0.294	0.0	0.368	0.419	0.189

Table D.80: Accuracy per prompt template for text-davinci-002-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	73.7	76.2	80.2	79.5	79.8	80.7
2	69.5	73.5	78.2	78.5	76.7	79.8
3	73.0	78.7	82.8	82.8	82.7	82.8
4	71.3	79.7	80.5	80.8	82.0	81.5
5	67.5	72.5	79.2	79.2	77.0	79.8
6	68.5	73.2	76.5	76.5	76.2	79.2
Mean	70.6	75.6	79.6	79.5	79.1	80.6
– std	2.28	2.79	1.96	1.94	2.6	1.22
Structured	72.7	78.2	81.2	81.0	81.5	81.7
– std	1.01	1.47	1.16	1.36	1.24	0.865
Natural	68.5	73.1	78.0	78.1	76.6	79.6
– std	0.816	0.419	1.11	1.14	0.33	0.283

Table D.81: Accuracy per prompt template for text-davinci-003-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	74.3	71.7	79.8	80.2	80.7	80.3
2	71.8	75.0	80.2	78.8	78.2	78.3
3	71.8	73.7	79.7	79.5	79.2	81.2
4	65.2	74.2	78.5	78.2	79.7	79.5
5	72.2	75.3	80.2	78.5	78.2	78.8
6	72.2	76.0	79.7	78.8	78.3	79.0
Mean	71.2	74.3	79.7	79.0	79.0	79.5
– std	2.84	1.38	0.57	0.666	0.929	0.975
Structured	70.4	73.2	79.3	79.3	79.9	80.3
– std	3.84	1.08	0.591	0.829	0.624	0.694
Natural	72.1	75.4	80.0	78.7	78.2	78.7
– std	0.189	0.419	0.236	0.141	0.0471	0.294

Table D.82: Accuracy per prompt template for ChatGPT-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	77.8	73.3	72.7	72.7	74.2	74.5
2	73.2	76.2	78.7	78.2	79.7	79.2
3	72.7	74.0	74.3	74.7	75.0	74.8
4	59.3	73.7	60.8	63.5	66.0	68.0
5	74.7	76.8	77.8	77.7	79.3	78.8
6	74.8	76.7	79.0	79.0	79.2	78.5
Mean	72.1	75.1	73.9	74.3	75.6	75.6
– std	5.94	1.48	6.29	5.29	4.79	3.9
Structured	69.9	73.7	69.3	70.3	71.7	72.4
– std	7.8	0.287	6.02	4.88	4.07	3.14
Natural	74.2	76.6	78.5	78.3	79.4	78.8
– std	0.732	0.262	0.51	0.535	0.216	0.287

Table D.83: Accuracy per prompt template for GPT-4-unknown.

Template	k = 0	k = 1	k = 5	k = 10	k = 15	k = 30
1	83.3	82.7	84.0	84.2	85.5	84.5
2	81.8	80.8	80.8	78.0	79.3	79.7
3	84.7	83.7	84.2	85.3	85.5	85.3
4	80.5	84.3	82.5	84.3	83.3	83.7
5	79.5	81.0	80.8	77.0	79.0	79.0
6	80.8	81.3	79.8	79.0	79.8	80.8
Mean	81.8	82.3	82.0	81.3	82.1	82.2
– std	1.76	1.36	1.67	3.37	2.81	2.44
Structured	82.8	83.6	83.6	84.6	84.8	84.5
– std	1.75	0.66	0.759	0.497	1.04	0.653
Natural	80.7	81.0	80.5	78.0	79.4	79.8
– std	0.942	0.205	0.471	0.816	0.33	0.741

D.8 Timestamps API calls

For reproducibility purposes, Table D.84, D.85, and D.86 contain the dates and times the APIs from OpenAI and Cohere were queried for the results.

Table D.84: Timestamp each was evaluated through OpenAI's API (1/2).

model	timestamp
GPT-3-ada/0-shot	2022-09-22 13:13:29
GPT-3-ada/1-shot	2022-09-22 15:11:13
GPT-3-ada/5-shot	2022-09-22 15:40:12
GPT-3-ada/10-shot	2022-09-22 18:14:18
GPT-3-ada/15-shot	2022-09-22 19:15:29
GPT-3-ada/30-shot	2022-09-22 22:47:58
GPT-3-babbage/0-shot	2022-09-22 23:19:05
GPT-3-babbage/1-shot	2022-09-22 23:39:53
GPT-3-babbage/5-shot	2022-09-23 00:01:32
GPT-3-babbage/10-shot	2022-09-23 00:24:27
GPT-3-babbage/15-shot	2022-09-23 00:49:13
GPT-3-babbage/30-shot	2022-09-23 01:15:44
GPT-3-curie/0-shot	2022-09-22 14:04:32
GPT-3-curie/1-shot	2022-09-23 02:09:14
GPT-3-curie/5-shot	2022-09-23 02:32:20
GPT-3-curie/10-shot	2022-09-23 02:56:43
GPT-3-curie/15-shot	2022-09-23 03:23:19
GPT-3-curie/30-shot	2022-09-23 03:52:30
GPT-3-davinci/0-shot	2022-09-22 12:21:48
GPT-3-davinci/1-shot	2022-09-23 14:27:15
GPT-3-davinci/5-shot	2022-09-23 15:10:40
GPT-3-davinci/10-shot	2022-09-23 16:04:53
GPT-3-davinci/15-shot	2022-09-23 17:17:04
GPT-3-davinci/30-shot	2022-09-23 18:36:38
OpenAI-text-ada-001/0-shot	2022-08-17 16:59:45
OpenAI-text-ada-001/1-shot	2022-08-17 18:23:12
OpenAI-text-ada-001/5-shot	2022-08-17 19:16:48
OpenAI-text-ada-001/10-shot	2022-08-17 20:24:16
OpenAI-text-ada-001/15-shot	2022-08-17 21:21:46
OpenAI-text-ada-001/30-shot	2022-08-17 22:44:47
OpenAI-text-babbage-001/0-shot	2022-08-17 11:50:44
OpenAI-text-babbage-001/1-shot	2022-08-17 12:22:08
OpenAI-text-babbage-001/5-shot	2022-08-17 12:50:59
OpenAI-text-babbage-001/10-shot	2022-08-17 13:27:52
OpenAI-text-babbage-001/15-shot	2022-08-17 14:57:43
OpenAI-text-babbage-001/30-shot	2022-08-17 15:45:16
OpenAI-text-curie-001/0-shot	2022-08-18 04:39:55
OpenAI-text-curie-001/1-shot	2022-08-18 05:10:17
OpenAI-text-curie-001/5-shot	2022-08-18 05:40:56
OpenAI-text-curie-001/10-shot	2022-08-18 06:15:28
OpenAI-text-curie-001/15-shot	2022-08-18 06:53:09
OpenAI-text-curie-001/30-shot	2022-08-18 07:35:40
OpenAI-text-davinci-001/0-shot	2022-08-26 20:26:21
OpenAI-text-davinci-001/1-shot	2022-08-26 21:02:31
OpenAI-text-davinci-001/5-shot	2022-08-26 21:35:19
OpenAI-text-davinci-001/10-shot	2022-08-27 07:14:02
OpenAI-text-davinci-001/15-shot	2022-08-27 07:58:25
OpenAI-text-davinci-001/30-shot	2022-08-27 08:44:42

Table D.85: Timestamp each was evaluated through OpenAI's API - continued (2/2).

model	timestamp
OpenAI-text-davinci-002/0-shot	2022-08-10 21:41:50
OpenAI-text-davinci-002/1-shot	2022-08-11 10:04:17
OpenAI-text-davinci-002/5-shot	2022-08-12 15:41:45
OpenAI-text-davinci-002/10-shot	2022-08-12 16:41:14
OpenAI-text-davinci-002/15-shot	2022-08-16 12:11:43
OpenAI-text-davinci-002/30-shot	2022-08-16 14:35:38
OpenAI-text-davinci-003/0-shot	2023-03-15 11:35:23
OpenAI-text-davinci-003/1-shot	2023-04-04 13:12:05
OpenAI-text-davinci-003/5-shot	2023-03-15 12:30:39
OpenAI-text-davinci-003/10-shot	2023-04-04 14:01:03
OpenAI-text-davinci-003/15-shot	2023-04-04 15:23:29
OpenAI-text-davinci-003/30-shot	2023-04-06 15:08:38
OpenAI-gpt-3.5.turbo/0-shot	2023-04-05 13:33:09
OpenAI-gpt-3.5.turbo/1-shot	2023-04-05 16:36:45
OpenAI-gpt-3.5.turbo/5-shot	2023-04-06 08:46:09
OpenAI-gpt-3.5.turbo/10-shot	2023-04-06 09:54:07
OpenAI-gpt-3.5.turbo/15-shot	2023-04-06 10:57:18
OpenAI-gpt-3.5.turbo/30-shot	2023-04-06 12:03:59
OpenAI-gpt-4/0-shot	2023-04-06 17:38:16
OpenAI-gpt-4/1-shot	2023-04-06 19:41:59
OpenAI-gpt-4/5-shot	2023-04-06 22:56:31
OpenAI-gpt-4/10-shot	2023-04-08 12:06:03
OpenAI-gpt-4/15-shot	2023-04-08 17:32:04
OpenAI-gpt-4/30-shot	2023-04-08 19:56:26

Table D.86: Timestamp each model was evaluated through Cohere's API.

model	timestamp
Cohere-small/0-shot	2022-08-16 22:22:17
Cohere-small/1-shot	2022-08-17 08:22:43
Cohere-small/5-shot	2022-08-17 09:19:57
Cohere-small/10-shot	2022-08-17 10:43:53
Cohere-small/15-shot	2022-08-17 12:53:02
Cohere-small/30-shot	2022-08-17 13:46:08
Cohere-medium/0-shot	2022-08-17 15:14:02
Cohere-medium/1-shot	2022-08-17 16:00:21
Cohere-medium/5-shot	2022-08-17 18:23:38
Cohere-medium/10-shot	2022-08-17 19:16:00
Cohere-medium/15-shot	2022-08-17 20:24:12
Cohere-medium/30-shot	2022-08-17 21:20:28
Cohere-large/0-shot	2022-08-17 22:47:49
Cohere-large/1-shot	2022-08-17 23:27:00
Cohere-large/5-shot	2022-08-18 00:10:08
Cohere-large/10-shot	2022-08-18 00:56:55
Cohere-large/15-shot	2022-08-18 01:48:30
Cohere-large/30-shot	2022-08-18 02:47:14
Cohere-xl/0-shot	2022-07-29
Cohere-xl/1-shot	2022-07-31
Cohere-xl/5-shot	2022-08-02
Cohere-xl/10-shot	2022-08-02 15:16:45
Cohere-xl/15-shot	2022-08-07 13:55:44
Cohere-xl/30-shot	2022-08-16 19:51:08
Cohere-command-medium/0-shot	2023-04-04 09:54:27
Cohere-command-medium/1-shot	2023-04-04 11:51:07
Cohere-command-medium/5-shot	2023-04-04 13:03:07
Cohere-command-medium/10-shot	2023-04-04 13:31:47
Cohere-command-medium/15-shot	2023-04-04 14:06:10
Cohere-command-medium/30-shot	2023-04-04 14:42:13
Cohere-command-xl/0-shot	2023-04-04 10:25:30
Cohere-command-xl/1-shot	2023-04-04 15:27:01
Cohere-command-xl/5-shot	2023-04-04 15:59:47
Cohere-command-xl/10-shot	2023-04-04 16:36:22
Cohere-command-xl/15-shot	2023-04-04 17:22:58
Cohere-command-xl/30-shot	2023-04-04 18:16:54

D.9 Compute and Emissions

Find below in Table D.87 until Table D.92 the timestamps, durations, and emissions per experiment (calculated with the CodeCarbon library in Python). Find below in Table D.93 until Table D.96 the cpu-type and count and gpu-type and count per experiment. In terms of compute the following GPU hours can be estimated if we assume each run is entirely done on the GPU (which is not true in reality, but worst case):

NVIDIA A100-SXM4-40GB used for 926.4291392151515 hours.

Tesla V100-PCIE-32GB used for 29.282544113265143 hours.

Tesla V100-PCIE-16GB used for 11.462701331244574 hours.

Table D.87: Timestamp, duration, and emissions per experiment with non-API models. (1/6)

model	timestamp	duration
EleutherAI-125m-0-shot	2022-09-01T21:33:14	8549.649220
EleutherAI-125m-1-shot	2022-09-02T00:10:03	640.861120
EleutherAI-125m-5-shot	2022-09-02T00:26:27	982.369876
EleutherAI-125m-10-shot	2022-09-02T00:51:24	1495.525381
EleutherAI-125m-15-shot	2022-09-02T01:29:03	2257.290708
EleutherAI-125m-30-shot	2022-09-02T09:04:03	27298.375266
EleutherAI-2.7b-0-shot	2022-09-03T00:36:14	3752.897449
EleutherAI-2.7b-1-shot	2022-09-03T02:04:16	5279.884696
EleutherAI-2.7b-5-shot	2022-09-03T04:28:19	8641.654516
EleutherAI-2.7b-10-shot	2022-09-03T08:18:13	13792.592126
EleutherAI-2.7b-15-shot	2022-09-03T13:33:25	18909.551123
EleutherAI-2.7b-30-shot	2022-09-03T22:47:06	33219.682098
EleutherAI-20b-0-shot	2022-08-25T07:40:55	1378.197924
EleutherAI-20b-1-shot	2022-08-25T08:15:23	807.702344
EleutherAI-20b-5-shot	2022-08-25T15:39:51	859.585535
EleutherAI-20b-10-shot	2022-08-25T16:18:50	1175.128651
EleutherAI-20b-15-shot	2022-08-25T16:47:30	1713.266182
EleutherAI-20b-30-shot	2022-08-25T17:45:28	3469.811664
EleutherAI-6b-0-shot	2022-08-24T22:29:30	1287.627453
EleutherAI-6b-1-shot	2022-08-24T23:22:30	1831.554774
EleutherAI-6b-5-shot	2022-08-25T00:16:57	3255.128955
EleutherAI-6b-10-shot	2022-08-25T01:23:21	3971.650578
EleutherAI-6b-15-shot	2022-08-25T02:26:23	3772.113814
EleutherAI-6b-30-shot	2022-08-25T04:18:30	6719.419030
EleutherAI-1.3b-0-shot	2022-09-02T09:54:06	3000.666020
EleutherAI-1.3b-1-shot	2022-09-02T10:46:30	3142.207699
EleutherAI-1.3b-5-shot	2022-09-02T12:25:25	5933.046596
EleutherAI-1.3b-10-shot	2022-09-02T12:39:00	8509.257493
EleutherAI-1.3b-15-shot	2022-09-02T18:00:39	11615.289366
EleutherAI-1.3b-30-shot	2022-09-02T23:33:39	19978.306457

Table D.88: Timestamp, duration, and emissions per experiment with non-API models. (2/6)

model	timestamp	duration
BLOOM-3b-0-shot	2022-08-31T12:54:37	5178.369790
BLOOM-3b-1-shot	2022-08-31T14:39:32	6292.560350
BLOOM-3b-5-shot	2022-08-31T17:37:29	10675.230701
BLOOM-3b-10-shot	2022-08-31T21:59:27	15715.744792
BLOOM-3b-15-shot	2022-09-01T03:41:02	20492.823278
BLOOM-3b-30-shot	2022-09-01T15:47:21	43577.882397
BLOOM-7b1-0-shot	2022-08-25T04:56:35	625.931470
BLOOM-7b1-1-shot	2022-08-25T05:07:13	630.628939
BLOOM-7b1-5-shot	2022-08-25T05:24:22	1022.138932
BLOOM-7b1-10-shot	2022-08-25T05:49:00	1471.008220
BLOOM-7b1-15-shot	2022-08-25T06:23:26	2058.455127
BLOOM-7b1-30-shot	2022-08-25T07:29:46	3972.772039
BLOOM-560m-0-shot	2022-08-29T15:35:52	2541.248956
BLOOM-560m-1-shot	2022-08-29T18:52:16	2532.794568
BLOOM-560m-5-shot	2022-08-29T20:16:16	5038.547060
BLOOM-560m-10-shot	2022-08-29T22:17:43	7285.239875
BLOOM-560m-15-shot	2022-08-30T00:38:23	8438.096533
BLOOM-560m-30-shot	2022-08-30T04:38:44	14419.447170
BLOOM-1b1-0-shot	2022-08-30T05:18:44	2398.828856
BLOOM-1b1-1-shot	2022-08-30T06:06:45	2879.435828
BLOOM-1b1-5-shot	2022-08-30T07:35:59	5352.607075
BLOOM-1b1-10-shot	2022-08-30T10:15:02	9541.535419
BLOOM-1b1-15-shot	2022-08-30T13:22:42	11257.077128
BLOOM-1b1-30-shot	2022-08-30T18:08:15	17131.797610
BLOOM-176b-0-shot	2022-10-14T12:51:11	3015.240235
BLOOM-176b-1-shot	2022-10-14T13:57:53	3906.461752
BLOOM-176b-5-shot	2022-10-14T20:41:10	7411.725385
BLOOM-176b-10-shot	2022-10-23T21:43:21	14462.201855
BLOOM-176b-15-shot	2022-10-24T01:14:10	12609.026736
BLOOM-176b-30-shot	2022-10-14T20:47:02	33159.499966

Table D.89: Timestamp, duration, and emissions per experiment with non-API models. (3/6)

model	timestamp	duration
OPT-13b-0-shot	2022-08-25T07:07:08	878.202579
OPT-13b-1-shot	2022-08-25T07:31:30	458.133617
OPT-13b-5-shot	2022-08-25T07:37:39	578.308507
OPT-13b-10-shot	2022-08-25T08:01:50	821.158826
OPT-13b-15-shot	2022-08-25T08:20:49	1131.479665
OPT-13b-30-shot	2022-08-25T16:05:27	2235.869414
OPT-350m-0-shot	2022-09-16T17:26:28	389.173905
OPT-350m-1-shot	2022-09-16T17:33:42	424.832551
OPT-350m-5-shot	2022-09-16T18:00:14	1583.824094
OPT-350m-10-shot	2022-09-16T18:32:12	1908.822462
OPT-350m-15-shot	2022-09-16T19:03:23	1863.625027
OPT-350m-30-shot	2022-09-16T19:47:29	2637.811867
OPT-125m-0-shot	2022-09-16T15:15:56	273.178967
OPT-125m-1-shot	2022-09-16T15:20:28	259.680856
OPT-125m-5-shot	2022-09-16T15:41:37	1259.801105
OPT-125m-10-shot	2022-09-16T16:09:59	1693.598805
OPT-125m-15-shot	2022-09-16T16:41:46	1899.415318
OPT-125m-30-shot	2022-09-16T17:19:51	2276.441314
OPT-6.7b-0-shot	2022-08-24T23:03:07	1140.485014
OPT-6.7b-1-shot	2022-08-24T23:17:51	872.225225
OPT-6.7b-5-shot	2022-08-24T23:34:40	995.894396
OPT-6.7b-10-shot	2022-08-24T23:55:44	1252.956499
OPT-6.7b-15-shot	2022-08-25T00:23:04	1627.749039
OPT-6.7b-30-shot	2022-08-25T01:05:49	2553.054289
OPT-2.7b-0-shot	2022-09-18T16:35:05	686.197892
OPT-2.7b-1-shot	2022-09-18T16:45:11	593.508211
OPT-2.7b-5-shot	2022-09-18T17:12:11	1613.313387
OPT-2.7b-10-shot	2022-09-18T17:44:48	1949.808232
OPT-2.7b-15-shot	2022-09-18T18:22:02	2225.927837
OPT-2.7b-30-shot	2022-09-18T19:09:05	2815.327871
OPT-30b-0-shot	2022-08-25T19:03:37	591.665447
OPT-30b-1-shot	2022-08-25T19:14:32	645.923823
OPT-30b-5-shot	2022-08-25T16:44:22	1825.821606
OPT-30b-10-shot	2022-08-25T17:07:22	1372.752916
OPT-30b-15-shot	2022-08-25T17:41:05	2015.006104
OPT-30b-30-shot	2022-08-25T18:10:39	3859.078056
OPT-1.3b-0-shot	2022-09-17T17:53:50	595.193443
OPT-1.3b-1-shot	2022-09-17T18:03:45	579.367790
OPT-1.3b-5-shot	2022-09-17T18:33:18	1759.103432
OPT-1.3b-10-shot	2022-09-17T19:12:19	2327.300123
OPT-1.3b-15-shot	2022-09-17T19:48:32	2161.637401
OPT-1.3b-30-shot	2022-09-17T20:37:00	2893.829010

Table D.90: Timestamp, duration, and emissions per experiment with non-API models. (4/6)

model	timestamp	duration
OPT-175b-0-shot	2022-10-19T15:02:56	2387.104187
OPT-175b-1-shot	2022-10-19T16:34:06	1589.972279
OPT-175b-5-shot	2022-10-19T17:25:58	3072.591171
OPT-175b-10-shot	2022-10-19T17:33:15	6211.692086
OPT-175b-15-shot	2022-10-19T21:29:16	8019.585246
OPT-175b-30-shot	2022-10-19T21:36:53	19901.470347
OPT-66b-0-shot	2022-08-25T18:58:11	2834.901372
OPT-66b-1-shot	2022-08-25T19:22:09	1427.806986
OPT-66b-5-shot	2022-08-25T19:47:39	1521.168440
OPT-66b-10-shot	2022-08-25T20:24:56	2228.407874
OPT-66b-15-shot	2022-08-25T21:41:21	3370.689256
OPT-66b-30-shot	2022-08-26T00:31:36	6816.312183

Table D.91: Timestamp, duration, and emissions per experiment with non-API models. (5/6)

model	timestamp	duration
BlenderBot-2.7b-0-shot	2022-09-04T08:09:56	3656.381540
BlenderBot-2.7b-1-shot	2022-09-12T15:58:01	4051.858183
BlenderBot-2.7b-5-shot	2022-09-12T17:16:20	4696.628979
BlenderBot-2.7b-10-shot	2022-09-12T18:35:53	4772.083818
BlenderBot-2.7b-15-shot	2022-09-12T19:54:13	4698.638356
BlenderBot-2.7b-30-shot	2022-09-12T21:10:34	4579.460884
BlenderBot-9.4b-0-shot	2022-10-22T04:04:24	614.201131
BlenderBot-9.4b-1-shot	2022-10-22T17:17:21	659.975971
BlenderBot-9.4b-5-shot	2022-10-22T17:31:48	839.336277
BlenderBot-9.4b-10-shot	2022-10-22T17:46:18	843.852691
BlenderBot-9.4b-15-shot	2022-10-22T17:53:41	1262.038660
BlenderBot-9.4b-30-shot	2022-10-22T18:23:25	853.334728
BlenderBot-90m-0-shot	2022-09-14T15:11:44	273.134700
BlenderBot-90m-1-shot	2022-09-14T15:17:38	351.542638
BlenderBot-90m-5-shot	2022-09-14T15:29:50	730.774348
BlenderBot-90m-10-shot	2022-09-14T15:47:22	1050.647882
BlenderBot-90m-15-shot	2022-09-14T16:07:27	1204.079804
BlenderBot-90m-30-shot	2022-09-14T16:28:55	1285.913686
T0-3b-0-shot	2022-10-21T17:33:36	348.245298
T0-3b-1-shot	2022-10-24T23:20:57	350.730799
T0-3b-5-shot	2022-10-24T23:29:21	474.378557
T0-3b-10-shot	2022-10-25T15:56:54	676.111759
T0-3b-15-shot	2022-10-25T16:12:55	928.215524
T0-3b-30-shot	2022-10-24T23:30:17	1961.897054
T0-11b-0-shot	2022-10-21T15:38:13	2289.815276
T0-11b-1-shot	2022-10-22T19:18:25	814.872760
T0-11b-5-shot	2022-10-22T19:41:45	1368.644314
T0-11b-10-shot	2022-10-22T20:17:30	2112.628515
T0-11b-15-shot	2022-10-22T21:06:30	2904.655213
T0-11b-30-shot	2022-10-22T22:41:16	5648.105648

Table D.92: Timestamp, duration, and emissions per experiment with non-API models. (6/6)

model	timestamp	duration
Flan-T5-3b-0-shot	2022-10-24T11:20:36	617.820384
Flan-T5-3b-1-shot	2022-10-25T12:29:59	348.405589
Flan-T5-3b-5-shot	2022-10-25T12:38:24	474.872964
Flan-T5-3b-10-shot	2022-10-25T12:50:00	665.592482
Flan-T5-3b-15-shot	2022-10-25T13:05:34	902.197151
Flan-T5-3b-30-shot	2022-10-25T13:37:14	1864.885266
Flan-T5-780m-0-shot	2022-10-24T11:54:09	160.503411
Flan-T5-780m-1-shot	2022-10-25T14:41:28	3816.321305
Flan-T5-780m-5-shot	2022-10-25T14:46:09	251.699700
Flan-T5-780m-10-shot	2022-10-25T14:52:09	331.340966
Flan-T5-780m-15-shot	2022-10-25T14:59:00	381.107934
Flan-T5-780m-30-shot	2022-10-25T15:11:18	705.711192
Flan-T5-11b-0-shot	2022-10-24T10:25:09	1111.283857
Flan-T5-11b-1-shot	2022-10-24T10:56:52	654.411412
Flan-T5-11b-5-shot	2022-10-25T17:26:50	1403.159768
Flan-T5-11b-10-shot	2022-10-25T18:29:59	3756.529085
Flan-T5-11b-15-shot	2022-10-25T19:21:15	3042.271478
Flan-T5-11b-30-shot	2022-10-25T20:57:13	5722.244579

Table D.93: Compute used per experiment with non-API models. (1/4)

model	cpus	cpu model	gpu model
EleutherAI-125m-0-shot	10	Apple M1 Max	
EleutherAI-125m-1-shot	10	Apple M1 Max	
EleutherAI-125m-5-shot	10	Apple M1 Max	
EleutherAI-125m-10-shot	10	Apple M1 Max	
EleutherAI-125m-15-shot	10	Apple M1 Max	
EleutherAI-125m-30-shot	10	Apple M1 Max	
EleutherAI-2.7b-0-shot	10	Apple M1 Max	
EleutherAI-2.7b-1-shot	10	Apple M1 Max	
EleutherAI-2.7b-5-shot	10	Apple M1 Max	
EleutherAI-2.7b-10-shot	10	Apple M1 Max	
EleutherAI-2.7b-15-shot	10	Apple M1 Max	
EleutherAI-2.7b-30-shot	10	Apple M1 Max	
EleutherAI-20b-0-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-20b-1-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-20b-5-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-20b-10-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-20b-15-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-20b-30-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-6b-0-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-6b-1-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-6b-5-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-6b-10-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-6b-15-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-6b-30-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
EleutherAI-1.3b-0-shot	10	Apple M1 Max	
EleutherAI-1.3b-1-shot	10	Apple M1 Max	
EleutherAI-1.3b-5-shot	10	Apple M1 Max	
EleutherAI-1.3b-10-shot	10	Apple M1 Max	
EleutherAI-1.3b-15-shot	10	Apple M1 Max	
EleutherAI-1.3b-30-shot	10	Apple M1 Max	

Table D.94: Compute used per experiment with non-API models. (2/4)

model	cpus	cpu model	gpu model
BLOOM-3b-0-shot	10	Apple M1 Max	
BLOOM-3b-1-shot	10	Apple M1 Max	
BLOOM-3b-5-shot	10	Apple M1 Max	
BLOOM-3b-10-shot	10	Apple M1 Max	
BLOOM-3b-15-shot	10	Apple M1 Max	
BLOOM-3b-30-shot	10	Apple M1 Max	
BLOOM-7b1-0-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
BLOOM-7b1-1-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
BLOOM-7b1-5-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
BLOOM-7b1-10-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
BLOOM-7b1-15-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
BLOOM-7b1-30-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
BLOOM-560m-0-shot	10	Apple M1 Max	
BLOOM-560m-1-shot	10	Apple M1 Max	
BLOOM-560m-5-shot	10	Apple M1 Max	
BLOOM-560m-10-shot	10	Apple M1 Max	
BLOOM-560m-15-shot	10	Apple M1 Max	
BLOOM-560m-30-shot	10	Apple M1 Max	
BLOOM-1b1-0-shot	10	Apple M1 Max	
BLOOM-1b1-1-shot	10	Apple M1 Max	
BLOOM-1b1-5-shot	10	Apple M1 Max	
BLOOM-1b1-10-shot	10	Apple M1 Max	
BLOOM-1b1-15-shot	10	Apple M1 Max	
BLOOM-1b1-30-shot	10	Apple M1 Max	
BLOOM-176b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BLOOM-176b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BLOOM-176b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BLOOM-176b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BLOOM-176b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BLOOM-176b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB

Table D.95: Compute used per experiment with non-API models. (3/4)

model	cpus	cpu model	gpu model
OPT-13b-0-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-13b-1-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-13b-5-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-13b-10-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-13b-15-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-13b-30-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-350m-0-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-350m-1-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-350m-5-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-350m-10-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-350m-15-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-350m-30-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-125m-0-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-125m-1-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-125m-5-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-125m-10-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-125m-15-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-125m-30-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-6.7b-0-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-6.7b-1-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-6.7b-5-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-6.7b-10-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-6.7b-15-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-6.7b-30-shot	1	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-2.7b-0-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-2.7b-1-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-2.7b-5-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-2.7b-10-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-2.7b-15-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-2.7b-30-shot	24	Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz	4 x Tesla V100-PCIE-32GB
OPT-30b-0-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-30b-1-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-30b-5-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-30b-10-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-30b-15-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-30b-30-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-1.3b-0-shot	40	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	4 x Tesla V100-PCIE-16GB
OPT-1.3b-1-shot	40	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	4 x Tesla V100-PCIE-16GB
OPT-1.3b-5-shot	40	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	4 x Tesla V100-PCIE-16GB
OPT-1.3b-10-shot	40	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	4 x Tesla V100-PCIE-16GB
OPT-1.3b-15-shot	40	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	4 x Tesla V100-PCIE-16GB
OPT-1.3b-30-shot	40	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	4 x Tesla V100-PCIE-16GB
OPT-175b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
OPT-175b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
OPT-175b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
OPT-175b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
OPT-175b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
OPT-175b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
OPT-66b-0-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-66b-1-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-66b-5-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-66b-10-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-66b-15-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB
OPT-66b-30-shot	48	Intel(R) Xeon(R) Platinum 8275CL CPU @ 3.00GHz	8 x NVIDIA A100-40GB

Table D.96: Compute used per experiment with non-API models. (4/4)

model	cpus	cpu model	gpu model
BlenderBot-2.7b-0-shot	10	Apple M1 Max	
BlenderBot-2.7b-1-shot	10	Apple M1 Max	
BlenderBot-2.7b-5-shot	10	Apple M1 Max	
BlenderBot-2.7b-10-shot	10	Apple M1 Max	
BlenderBot-2.7b-15-shot	10	Apple M1 Max	
BlenderBot-2.7b-30-shot	10	Apple M1 Max	
BlenderBot-9.4b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BlenderBot-9.4b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BlenderBot-9.4b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BlenderBot-9.4b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BlenderBot-9.4b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BlenderBot-9.4b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
BlenderBot-90m-0-shot	10	Apple M1 Max	
BlenderBot-90m-1-shot	10	Apple M1 Max	
BlenderBot-90m-5-shot	10	Apple M1 Max	
BlenderBot-90m-10-shot	10	Apple M1 Max	
BlenderBot-90m-15-shot	10	Apple M1 Max	
BlenderBot-90m-30-shot	10	Apple M1 Max	
T0-3b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-3b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-3b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-3b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-3b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-3b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-11b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-11b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-11b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-11b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-11b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
T0-11b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-3b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-3b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-3b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-3b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-3b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-3b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-780m-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-780m-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-780m-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-780m-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-780m-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-780m-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-11b-0-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-11b-1-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-11b-5-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-11b-10-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-11b-15-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB
Flan-T5-11b-30-shot	96	Intel(R) Xeon(R) CPU @ 2.20GHz	16 x NVIDIA A100-40GB

Appendix E

A Case Study in Social Reasoning: Theory of Mind

This chapter contains the raw counts for the experiments in Chapter 6.

E.1 Detailed results

Table E.1 and E.2 show the results for GPT-4 on the pillar target templates and fruit target templates respectively. Table E.3 and E.4 show the results for GPT-3.5-turbo on the pillar target templates and fruit target templates respectively. Table E.5 and E.6 show the results for text-davinci-003 on the pillar target templates and fruit target templates respectively.

Table E.1: Animate, inanimate, and control object and location bias for GPT-4 on prompts from the group Pillar targets. H stands for habituations, and Anim for whether (Y) or not (N) the prompt template has animate denotation.

Category	Model	H	Anim	N obj bias	N loc bias	N unclassified
Animate	GPT-4	0	N	3.8 +/- 4.5	3.0 +/- 4.3	3.2 +/- 4.4
	GPT-4	6	N	0.4 +/- 1.4	9.6 +/- 1.4	0.0 +/- 0.1
	GPT-4	0	Y	3.7 +/- 4.3	3.2 +/- 4.0	3.0 +/- 3.9
	GPT-4	6	Y	0.5 +/- 1.6	9.5 +/- 1.6	0.0 +/- 0.0
Inanimate	GPT-4	0	N	3.6 +/- 4.5	3.4 +/- 4.4	3.1 +/- 4.3
	GPT-4	6	N	0.1 +/- 0.7	9.9 +/- 0.7	0.0 +/- 0.0
	GPT-4	0	Y	3.6 +/- 4.6	3.2 +/- 4.5	3.3 +/- 4.5
	GPT-4	6	Y	0.0 +/- 0.1	10.0 +/- 0.1	0.0 +/- 0.1
Control	GPT-4	0	N	3.6 +/- 3.9	3.2 +/- 3.5	3.2 +/- 3.6
	GPT-4	6	N	0.1 +/- 0.3	9.9 +/- 0.3	0.0 +/- 0.0
	GPT-4	0	Y	3.7 +/- 3.5	3.3 +/- 3.4	3.0 +/- 3.4
	GPT-4	6	Y	0.0 +/- 0.0	10.0 +/- 0.0	0.0 +/- 0.0

Table E.2: Animate, inanimate, and control object and location bias for GPT-4 on prompts from the group Fruit targets. H stands for habituations, and Anim for whether (Y) or not (N) the prompt template has animate denotation.

Category	Model	H	Anim	N obj bias	N loc bias	N unclassified
Animate	GPT-4	0	N	0.0 +/- 0.0	0.0 +/- 0.1	10.0 +/- 0.1
	GPT-4	6	N	6.1 +/- 3.7	2.0 +/- 3.2	1.8 +/- 3.0
	GPT-4	0	Y	0.0 +/- 0.0	0.0 +/- 0.0	10.0 +/- 0.0
	GPT-4	6	Y	5.4 +/- 3.8	1.0 +/- 2.4	3.6 +/- 3.7
Inanimate	GPT-4	0	N	0.0 +/- 0.0	0.0 +/- 0.0	10.0 +/- 0.0
	GPT-4	6	N	3.0 +/- 3.5	5.7 +/- 4.1	1.3 +/- 2.5
	GPT-4	0	Y	0.0 +/- 0.1	0.0 +/- 0.1	10.0 +/- 0.2
	GPT-4	6	Y	2.3 +/- 3.1	5.8 +/- 4.0	1.9 +/- 3.1
Control	GPT-4	0	N	0.0 +/- 0.2	0.0 +/- 0.2	9.9 +/- 0.3
	GPT-4	6	N	5.5 +/- 3.5	1.6 +/- 2.9	2.9 +/- 3.1
	GPT-4	0	Y	0.0 +/- 0.0	0.0 +/- 0.0	10.0 +/- 0.0
	GPT-4	6	Y	5.0 +/- 3.6	2.6 +/- 3.7	2.4 +/- 3.0

Table E.3: Animate, inanimate, and control object and location bias for GPT-3.5-turbo on prompts from the group Pillar targets. H stands for habituations, and Anim for whether (Y) or not (N) the prompt template has animate denotation.

Category	Model	H	Anim	N obj bias	N loc bias	N unclassified
Animate	GPT-3.5-turbo	0	N	3.7 +/- 4.4	2.7 +/- 4.0	3.6 +/- 4.4
	GPT-3.5-turbo	6	N	4.5 +/- 4.3	5.3 +/- 4.2	0.2 +/- 0.6
	GPT-3.5-turbo	0	Y	3.2 +/- 4.4	2.7 +/- 4.2	4.1 +/- 4.6
	GPT-3.5-turbo	6	Y	5.2 +/- 4.5	4.7 +/- 4.5	0.1 +/- 0.5
Inanimate	GPT-3.5-turbo	0	N	3.2 +/- 3.7	3.0 +/- 3.6	3.9 +/- 3.8
	GPT-3.5-turbo	6	N	3.3 +/- 3.7	6.0 +/- 3.8	0.7 +/- 1.5
	GPT-3.5-turbo	0	Y	3.1 +/- 4.2	2.9 +/- 4.0	4.0 +/- 4.3
	GPT-3.5-turbo	6	Y	3.3 +/- 3.9	6.2 +/- 3.9	0.5 +/- 1.2
Control	GPT-3.5-turbo	0	N	3.1 +/- 4.4	2.9 +/- 4.4	4.0 +/- 4.7
	GPT-3.5-turbo	6	N	3.7 +/- 4.4	5.3 +/- 4.3	0.9 +/- 2.2
	GPT-3.5-turbo	0	Y	3.2 +/- 4.4	2.8 +/- 4.1	4.1 +/- 4.6
	GPT-3.5-turbo	6	Y	3.8 +/- 4.4	6.0 +/- 4.4	0.2 +/- 0.7

Table E.4: Animate, inanimate, and control object and location bias for GPT-3.5-turbo on prompts from the group Fruit targets. H stands for habituations, and Anim for whether (Y) or not (N) the prompt template has animate denotation.

Category	Model	H	Anim	N obj bias	N loc bias	N unclassified
Animate	GPT-3.5-turbo	0	N	0.1 +/- 0.3	0.1 +/- 0.4	9.8 +/- 0.4
	GPT-3.5-turbo	6	N	6.0 +/- 3.4	1.8 +/- 3.2	2.2 +/- 2.3
	GPT-3.5-turbo	0	Y	0.1 +/- 0.3	0.1 +/- 0.3	9.9 +/- 0.4
	GPT-3.5-turbo	6	Y	7.4 +/- 2.6	0.8 +/- 2.2	1.8 +/- 2.0
Inanimate	GPT-3.5-turbo	0	N	0.3 +/- 0.6	0.2 +/- 0.7	9.5 +/- 0.9
	GPT-3.5-turbo	6	N	6.3 +/- 3.8	2.8 +/- 3.6	0.8 +/- 1.5
	GPT-3.5-turbo	0	Y	0.9 +/- 1.4	0.7 +/- 1.1	8.4 +/- 1.6
	GPT-3.5-turbo	6	Y	5.3 +/- 4.2	4.0 +/- 4.3	0.7 +/- 1.4
Control	GPT-3.5-turbo	0	N	0.0 +/- 0.2	0.0 +/- 0.2	9.9 +/- 0.2
	GPT-3.5-turbo	6	N	7.8 +/- 3.1	1.6 +/- 3.2	0.6 +/- 1.0
	GPT-3.5-turbo	0	Y	0.0 +/- 0.2	0.1 +/- 0.2	9.9 +/- 0.3
	GPT-3.5-turbo	6	Y	8.2 +/- 2.8	1.3 +/- 2.8	0.5 +/- 0.8

Table E.5: Animate, inanimate, and control object and location bias for text-davinci-003 on prompts from the group Pillar targets. H stands for habituations, and Anim for whether (Y) or not (N) the prompt template has animate denotation.

Category	Model	H	Anim	Obj p	Loc p	Obj bias
Animate	text-davinci-003	0	N	0.3 +/- 0.2	0.3 +/- 0.2	0.5 +/- 0.2
	text-davinci-003	6	N	0.0 +/- 0.1	1.0 +/- 0.1	0.0 +/- 0.1
	text-davinci-003	0	Y	0.3 +/- 0.2	0.3 +/- 0.2	0.5 +/- 0.2
	text-davinci-003	6	Y	0.0 +/- 0.1	1.0 +/- 0.1	0.0 +/- 0.1
Inanimate	text-davinci-003	0	N	0.1 +/- 0.1	0.1 +/- 0.1	0.5 +/- 0.2
	text-davinci-003	6	N	0.0 +/- 0.0	1.0 +/- 0.0	0.0 +/- 0.0
	text-davinci-003	0	Y	0.2 +/- 0.1	0.2 +/- 0.1	0.5 +/- 0.2
	text-davinci-003	6	Y	0.0 +/- 0.0	1.0 +/- 0.0	0.0 +/- 0.0
Control	text-davinci-003	0	N	0.3 +/- 0.2	0.3 +/- 0.2	0.5 +/- 0.2
	text-davinci-003	6	N	0.0 +/- 0.0	1.0 +/- 0.0	0.0 +/- 0.0
	text-davinci-003	0	Y	0.3 +/- 0.2	0.3 +/- 0.2	0.5 +/- 0.2
	text-davinci-003	6	Y	0.0 +/- 0.0	1.0 +/- 0.0	0.0 +/- 0.0

Table E.6: Animate, inanimate, and control object and location bias for text-davinci-003 on prompts from the group Fruit targets. H stands for habituations, and Anim for whether (Y) or not (N) the prompt template has animate denotation.

Category	Model	H	Anim	Obj p	Loc p	Obj bias
Animate	text-davinci-003	0	N	0.1 +/- 0.1	0.1 +/- 0.1	0.5 +/- 0.2
	text-davinci-003	6	N	0.9 +/- 0.2	0.1 +/- 0.2	0.9 +/- 0.2
	text-davinci-003	0	Y	0.1 +/- 0.1	0.1 +/- 0.1	0.5 +/- 0.2
	text-davinci-003	6	Y	1.0 +/- 0.2	0.0 +/- 0.2	1.0 +/- 0.2
Inanimate	text-davinci-003	0	N	0.1 +/- 0.0	0.1 +/- 0.0	0.5 +/- 0.2
	text-davinci-003	6	N	0.7 +/- 0.4	0.3 +/- 0.4	0.7 +/- 0.4
	text-davinci-003	0	Y	0.0 +/- 0.0	0.0 +/- 0.0	0.5 +/- 0.2
	text-davinci-003	6	Y	0.8 +/- 0.3	0.2 +/- 0.3	0.8 +/- 0.3
Control	text-davinci-003	0	N	0.0 +/- 0.0	0.0 +/- 0.0	0.5 +/- 0.3
	text-davinci-003	6	N	1.0 +/- 0.0	0.0 +/- 0.0	1.0 +/- 0.0
	text-davinci-003	0	Y	0.1 +/- 0.0	0.1 +/- 0.0	0.5 +/- 0.2
	text-davinci-003	6	Y	1.0 +/- 0.0	0.0 +/- 0.1	1.0 +/- 0.0