Structure-Guided Approaches for Robust Language Model Reasoning

Melanie Sclar

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2026

Reading Committee:

Yejin Choi, Co-Chair
Yulia Tsvetkov, Co-Chair
Luke Zettlemoyer
Max Kleiman-Weiner

Program Authorized to Offer Degree:

Paul G. Allen School of Computer Science & Engineering

University of Washington

Abstract

Structure-Guided Approaches for Robust Language Model Reasoning

Melanie Sclar

Co-Chairs of the Supervisory Committee:

Yejin Choi
Paul G. Allen School of Computer Science & Engineering

Yulia Tsvetkov
Paul G. Allen School of Computer Science & Engineering

Large Language Models exhibit increasingly sophisticated behaviors, yet we lack systematic methods to evaluate their true reasoning capabilities. Current approaches—such as collecting human-generated data—cannot reliably predict model failure modes, distinguish genuine reasoning from complex memorization patterns, nor guarantee synthetic data quality.

This thesis explores the idea that imbuing tasks with explicit structure provides the control and verification mechanisms necessary for more principled evaluation and reliable data generation. Structure unlocks optimization and search algorithms that can systematically discover model flaws while supporting both higher-quality training data generation and

targeted inference-time improvements through more interpretable methods. This creates a virtuous cycle where better evaluation can lead to stronger modeling.

Throughout four works, I demonstrate this approach across degrees of structural abstraction, ranging from surface-level to semantic-level control. Applications span from prompt formatting sensitivity and compositional reasoning analysis, to theory of mind reasoning improvement and evaluation. Crucially, it is this explicit structure that unlocks diverse techniques including multi-armed bandits for exploration, A* search for adversarial discovery of challenging data, and custom graph algorithms for reasoning representation and mental state modeling.

By simultaneously showing successful applications for improving evaluation rigor, data quality, and overall model performance, structure-guided approaches offer a path toward more reliable and capable language modeling.

Para mi mamá Mónica, porque estamos siempre juntas.

Acknowledgments

I am deeply grateful to my advisors Yulia Tsvetkov and Yejin Choi for guiding me throughout my PhD, for taking a chance on me when I had extremely limited research experience, and for their amazing and always complementary research insights.

To Yulia, thank you for encouraging me to believe in myself and for giving me the freedom to pursue the directions I believed in most. You offered guidance along the way while trusting me with the space to think critically and grow into an independent researcher—the best result of that is what became Part I of this thesis. Your warmth and care made me feel like more than just a student in your lab, and you taught me that academic excellence and personal well-being can go hand in hand.

To Yejin, thank you for teaching me by example to keep dreaming bigger, to find new horizons, and to rapidly adapt to changing times. Thank you for teaching me to be stronger and communicate my research more clearly and compellingly. I have often found your advice in my head when I write any kind of text and when I present my work. As we will cover in Part III, having a robust theory of mind is important, and you taught me to sharpen mine.

To my committee members Luke Zettlemoyer and Max Kleiman-Weiner, thank you for your thoughtful questions and support throughout the defense process.

I was fortunate to work with some truly wonderful collaborators and mentors. To Alane Suhr, thank you for being my prototypical example of a great academic: you have the breadth of knowledge, the intellectual curiosity, and the patience that I aspire to have. You're also incredibly fun to be around! To Sachin Kumar and Peter West, who were my mentors at the beginning of my PhD and patiently helped me grow as a researcher. To Ximing Lu, the hardest-working researcher I've ever seen. To my many other collaborators: Zaid Harchaoui, Hyunwoo Kim, Kabir Ahuja, Shuyue Stella Li, Bill Yuchen Lin, Linlu Qiu, Skyler Hallinan, Sean Welleck, and more.

Thanks to all my tsvetshop and xlab labmates for the shared moments and discussions, including (in no particular order) Vidhisha, Chan, Han, Lucille, Abhilasha, Valentina and Yanai, Niloofar, Jillian, Taylor, Alisa, Ben, Etash, Jaehun, Liwei, and more.

I am thankful for the two years I spent in the Meta AIM program, which gave me a much deeper understanding of what being a research scientist entails. Thank you to my mentor Asli Celikyilmaz, and to my collaborators Jane Yu, Maryam Fazel-Zarandi, Ansong Ni, Andrew Cohen, and Bhargavi Paranjape.

A PhD wasn't the most predictable path from where I started in life, and I'm here also

thanks to all the people who believed in me, and thanks to the support I received since I was young in Argentina through free education and many scholarships. I would especially like to thank Yonatan Bisk and Graham Neubig for hosting me in my first academic ML research experience at CMU, and for supporting me through the craziness that an internship during COVID entailed. To Kilian Weinberger, for his generosity in letting me join some of his research projects at ASAPP. To Greg Diuk, for his encouragement and for writing, along with Kilian, the letters that got me here.

I've never missed a chance to criticize the Seattle freeze, but Seattle has nonetheless given me incredible friendships. To Inna Lin and Oreva Ahia, for lifting me up, for pushing me to prioritize my needs, and for always telling me what I needed to hear. To Weijia Shi, for being good cop when Inna and Oreva are being bad cop. To Stella Li, for bringing the fun wherever you go. To John Hewitt and Amanda Bertsch, for teaching me to distrust Idaho tacos. To Julian Reis, for telling me to *just go up* and being patient when I don't. Thanks also to Carolina, Esteban, Tim, Ivan, Chris Rytting, Hamish, KJ, Akari, Lancelot, Gian Marco, Liz, and the many others who have been with me through different moments of these four and a half years at UW. Thanks to all the friends who made conferences some of my favorite moments, and who I hope to keep running into forever (shoutout to Vishakh and Rheeya!).

Meta also made me meet some of my favorite people. To Baptiste, for brightening the last stretch of this PhD journey. To Tomasz, for all our deep and wide-ranging chats. To Mike, Arti, Hunter, Thao, Julie, Naman, Avi, Ari, Margaret, and everyone who saw my day radically improve after finding out there's Basque cheesecake for dessert.

To LLMs for never giving us a dull moment.

Living abroad can sometimes be hard. Thank you to the Argentinian++ crowd in Seattle for making it a little easier, and especially to Ale Candioti, Julija, Brian, Eunice, and to Mati Domingues for letting a stranger crash your house for 2+ weeks. To that place in Capitol Hill that had the best and only fugazzettas in town.

Thank you to my family. To my dad Carlos, my uncles Jorge and César, and my grandma Adela. To Carlitos and María, and to my cousins Cecilia, Virginia, and Florencia, for being there when I needed you most. To my mom Mónica and my abueli Amalia, who are always in my heart.

I've jokingly been saying that I've had several technical *eras*. None of this would have been possible without the first of all: math olympiads. Thank you to everyone at the Olimpíada Matemática Argentina for showing me the joy in problem-solving and teaching me how to persevere. Thank you to my lifelong friends Matías Saucedo, Sebastián Prillo, Agustín Santiago Gutiérrez, Miguel Maurizio, Ariel Zylber, Martín Mereb, among many others, who were always present despite the distance. Thanks to Nuria Madrid de Susmel for your continued mentorship: as she always says, the olympiads teach you to think about *"qué puedo hacer con lo que tengo?"*—in essence, "what can I achieve with the hypotheses I am given?". I've since been learning to question my hypotheses.

The PhD has been a growth era. On to the next one!

Contents

Chapter 1

Introduction

Early natural language processing systems were built around explicit representations of language and task structure: rule-based grammars, semantic formalisms, structured prediction. The past decade has seen a dramatic departure from this tradition, as transformer architectures and self-supervised pre-training have produced models that learn from vast quantities of unstructured text, achieving remarkable performance without any explicit encoding of task structure or linguistic knowledge. Without explicit structure to anchor our analyses or interpretability mechanisms to inspect learned representations, rigorously quantifying model behavior has become a fundamental challenge.

Large Language Models (LLMs) are being deployed in real-world applications at the same time that model-generated data increasingly becomes a core component of training. This makes quality assurance for LLM generations essential for both end users and model developers. However, current approaches, such as human-generated evaluation data, are often insufficient because they cannot reliably predict failure modes, hindering our ability to distinguish genuine reasoning from sophisticated memorization patterns. Moreover, poor control during text generation and evaluation also impacts training, as developers increasingly use synthetic training data.

Addressing these evaluation and generation challenges requires algorithmic solutions that can systematically explore the space of possible prompts and text generations, as well as reliably guide text generation. We argue that imbuing tasks with explicit structure provides the control and verification mechanisms necessary for principled LLM evaluation by unlocking optimization and search algorithms otherwise unsuitable for unstructured data. Crucially, the same structures that enable the systematic discovery of model failures often enable the systematic generation of higher quality training data with more interpretable generation algorithms. This creates a virtuous cycle where better evaluation can lead to stronger modeling. Structural control can also be applied as part of inference-time algorithms to improve reasoning capabilities for domains of interest in interpretable ways.

Concretely, we will demonstrate how such structured formulations—at the *surface* level, controlling how information is presented to the model, and at the *semantic* level, controlling what information is presented—can systematically discover model failures, generate high-quality training data, and improve inference-time reasoning, across different levels of abstraction and reasoning domains.

At the most surface level, model outputs are highly sensitive to semantically irrelevant

aspects of inputs (Lu et al., 2022; Mizrahi et al., 2024, inter alia): small, arbitrary choices in how a prompt is formatted can dramatically alter model behavior despite carrying no meaningful change in intent, leading to performance estimates that are confounded by choices that are rarely reported and poorly understood.

Moving deeper, while LLMs demonstrate striking capabilities across a wide range of reasoning tasks (Brown et al., 2020; Chowdhery et al., 2023; OpenAI, 2023), the same models fail surprisingly on problems that appear simple (Bian et al., 2024; Koralus and Wang-Mascianica, 2023; C. Qin et al., 2023). Benchmark failures alone are insufficient—what is needed are systematic methods for understanding *why* models fail, not just *that* they fail, so as to distinguish fundamental limitations from fixable training deficiencies. Conversely, benchmark successes are insufficient too: it is often unclear whether a model that succeeds has genuinely learned a reasoning capability, or whether it has memorized patterns that happen to pass within the evaluation examples considered. Formulating tasks with explicit structure—for instance, as computation graphs over discrete reasoning steps—makes complexity formally quantifiable and intermediate sub-procedures independently verifiable, enabling principled diagnosis of *why* and *where* models break down, rather than reducing evaluation to aggregate benchmark scores.

This difficulty in reaching scientific consensus on whether models possess a given capability is perhaps most acute for social reasoning, a core competency for any model that aspires to effectively interact with humans. The case of Theory of Mind (ToM) (Premack and Woodruff, 1978)—the capacity to reason about others' mental states, beliefs, and intentions, which underlies reading comprehension, dialogue, negotiation, and virtually every form of collaborative interaction—is illustrative: evidence for reliable theory of mind in LLMs has been the target of controversy, with studies alternately claiming that LLMs may possess (Bubeck et al., 2023; Kosinski, 2024) or lack (Sap et al., 2022; Shapira et al., 2024; Strachan et al., 2024; Ullman, 2023) this capability. These conflicting results are driven in part by benchmark limitations, as existing datasets are simplistic, often cumbersome to expand, and often prone to being solved through spurious surface cues rather than genuine belief tracking (Le, Boureau, and Nickel, 2019; Ullman, 2023). As we will show, imposing explicit structure on theory of mind tasks cuts through this impasse from both directions: it enables the algorithmic generation of adversarial scenarios that expose model failures with precision, and it supports both inference-time improvements that require no retraining and fine-tuning pipelines that yield substantial gains on established benchmarks.

By demonstrating successful applications across evaluation rigor, data quality, and model performance—at both the surface and semantic levels—this thesis argues that structure-guided approaches offer a principled path toward more reliable and capable language modeling.

1.1 Thesis Statement

Imbuing language model reasoning tasks with explicit structural constraints—spanning surface-level prompt structure and semantic-level reasoning domains such as compositional reasoning and theory of mind—unlocks principled evaluation, higher-quality data generation, and interpretable inference-time improvements that are otherwise unattainable with

unstructured approaches, creating a virtuous cycle in which better evaluation leads to stronger modeling.

1.2 Thesis Overview

The works below encompass different levels of abstraction in structure modeling and have impacts across many stages of language modeling development.

Part I shows how imposing structure over surface-level variations in prompts—variations that do not alter semantics, but may nonetheless affect model behavior—enables principled model analysis and evaluation. Prompt formatting is a canonical example of such variation: choices like separators, casing, and field ordering are unimportant to human readers yet can dramatically shift model outputs. By defining a grammar of valid prompt formats to systematically explore these variations, I implicitly define an equivalence class of semantically equivalent formats and cast the problem of estimating performance ranges across this class as a multi-armed bandit, enabling efficient exploration without accessing model weights (Chapter 2). The resulting tool has been adopted by other researchers as a practical prompt engineering method, extending its impact to inference-time performance improvements beyond its primary contribution to analysis and evaluation.

Turning to semantic-level structure, Parts II and III show how this deeper form of structural control can advance analysis and evaluation, data generation, and inference-time enhancement across two markedly different reasoning domains. Part II demonstrates how structure can be used to rigorously characterize the limits of LLM reasoning—an impact area that requires moving beyond aggregate benchmark scores toward principled, reproducible understanding of *why* models fail. Multi-step compositional reasoning, where problems must be broken into sub-steps and synthesized into precise answers, offers a particularly clean testbed: by formulating such tasks as computation graphs, complexity becomes formally quantifiable and reasoning steps can be broken into intermediate sub-procedures that are independently verifiable. Empirical findings suggest that transformers solve these tasks by reducing multi-step reasoning into linearized subgraph matching without necessarily developing systematic problem-solving skills; theoretical arguments further prove how zero-shot autoregressive performance provably decay exponentially with increased task complexity (Chapter 3).

While Part II demonstrates that structure-guided analysis can yield rigorous insights into model limitations, its focus on formal computational tasks leaves open the question of whether the same structural approach can be extended to domains requiring social rather than mathematical reasoning, and whether the goal can expand from evaluation alone to actively improving model capabilities. Part III takes up this challenge through theory of mind reasoning—the ability to understand others' mental states, beliefs, and intentions—which is key for social intelligence and prone to capability overestimation in LLMs. Semantic-level structure here enables advances across all three impact areas: analysis and evaluation, data generation, and inference-time enhancement.

For inference-time enhancement and evaluation, I introduce a plug-and-play multi-character belief tracker that reconstructs the latent structure present in false-belief theory of mind tasks, maintaining explicit symbolic representations of each character's beliefs

Impact Area	Structure Modeling by Abstraction Level	
	Surface	Semantic
Data Generation	• ExploreToM *(story structures vs. infilled stories)*	• Faith and Fate *(synthetically generated tasks, fine-tuning in some analyses)* • ExploreToM
Analysis and Evaluation	• FormatSpread	• Faith and Fate *(analyses)* • SymbolicToM *(benchmark brittleness analysis)* • ExploreToM
Inference-Time Enhancements	• FormatSpread *(used by other researchers as a prompt engineering method)*	• SymbolicToM

Table 1.1: Research contributions across structural approaches and impact areas

throughout a narrative to enable more reliable inference without retraining the underlying language model (Chapter 4). With the insight that structured task representations are both possible and beneficial for theory of mind, I then turn to data generation: I define a structured action space representing a small world model with associated mental state updates for each action, then leverage A* search over this domain-specific language to generate diverse, challenging theory of mind scenarios for both robust model evaluation and improvement through supervised fine-tuning (Chapter 5). Structure can coexist with diversity: by leveraging LLMs to sample diverse actions corresponding to the same mental state and world state updates, the resulting datasets achieve radical diversity without sacrificing formal verifiability.

All of these works have been published in major machine learning and natural language processing venues. In summary, the works covered are as follows:

I. Modeling Surface Structure

- **FormatSpread**: Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design (Sclar et al., 2024) (Chapter 2, ICLR 2024)

II. Modeling Semantic Structure for Compositional Reasoning Representation and Analysis

- **Faith and Fate**: Faith and Fate: Limits of Transformers on Compositionality (Dziri*, Lu*, Sclar* et al. 2023) (Chapter 3, NeurIPS 2023, Spotlight)

III. Modeling Semantic Structure: Theory of Mind Reasoning

- **SymbolicToM**: Minding Language Models' (Lack of) Theory of Mind: A Plug-and-Play Multi-Character Belief Tracker (Sclar et al., 2023) (Chapter 4, ACL 2023, Outstanding Paper Award)
- **ExploreToM**: Explore Theory of Mind: Program-Guided Adversarial Data Generation for Theory of Mind Reasoning (Sclar et al., 2025) (Chapter 5, ICLR 2025)

Each work's impact areas and structural modeling abstraction levels are shown in Table 1.1.

Part I

Modeling Surface Structure

Chapter 2

Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design

*This chapter is based on **FormatSpread**: Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. Published at ICLR 2024.*

Reliable evaluation of language model capabilities requires that performance estimates reflect genuine reasoning rather than incidental properties of how inputs are presented. Yet even before considering the semantic content of a task, a more basic source of confounding demands attention: the surface form of the prompt itself. Formatting choices—separators, casing, field ordering—are rarely reported in research papers, under the implicit assumption that they are inconsequential. This chapter challenges that assumption. By defining a *grammar* of semantically equivalent prompt formats, we give surface-level structure a precise role: it delineates an equivalence class of inputs that should, in principle, elicit identical model behavior, and we measure how far empirical reality departs from that ideal. Efficiently exploring this equivalence class is cast as a multi-armed bandit problem, yielding a principled estimate of performance variance that requires no access to model weights and scales to a user-specified computational budget.

The resulting evidence—that prompt formatting introduces substantial, largely unreported variance in model performance—is the first concrete instantiation of the thesis's central argument. Imposing structure on what is typically treated as an unstructured design space immediately unlocks algorithmic tools for principled analysis. Parts II and III will later extend this philosophy inward, from the surface form of prompts to the semantic structure of the reasoning tasks themselves, asking what richer structural representations can reveal about model limitations and enable in terms of data generation and inference-time enhancement.

2.1 Introduction

As the capabilities of LLMs have rapidly improved, their sensitivity to input prompt features has been used to optimize performance via prompt engineering (White et al., 2023). However, there has been little work in characterizing this sensitivity, especially to seemingly innocuous feature choices that preserve prompt meaning and intent. In this work, we analyze the sensitivity of widely used, open-source LLMs to a class of features that should not influence a prompt's interpretation: formatting choices. We find that pre-trained LLMs are sensitive to these choices in unpredictable ways, with accuracy varying in up to 76 points for LLaMA-2-13B between equivalent formats, and ∼10 accuracy points on average across 50+ tasks and several models. We also show that this variance is not eliminated by adding few-shot examples, increasing model size, or instruction tuning.

Designing prompt templates is a critical part of effectively using a pre-trained language model. This design process includes making choices about wording, choosing few-shot examples for in-context learning, and making decisions about seemingly trivial features like formatting. This process, and often even the resulting templates, is rarely reported or discussed in research papers, under the assumption that performance variance across these choices is insignificant compared to variance across data points or models. However, some anecdotal evidence points to formatting choices actually having a significant influence on model behavior (Aghajanyan, 2023). In some cases, researchers report a limited number of manually generated formats to show that scaling trends hold despite performance being significantly different (Schick, Udupa, and Schütze, 2021). The assumption that formatting does not influence overall model performance may become problematic when improvements over existing approaches are attributed to the amount and source of training data, number of parameters, or model architecture, without also accounting for changes in prompt format. Ignoring variance across formats may also negatively affect user experience, e.g. if users inadvertently choose formats the LLM does not perform well on.

Our proposed tool, FormatSpread, enables a systematic analysis of these variances across a wide set of semantically equivalent prompt formats within a user-specified computational budget. We find that choices in formatting few-shot examples during in-context learning introduce spurious biases that may lead to significantly different conclusions in model performance. The sensitivity to formatting choices that we discover across widely-used, open-source models suggests that future research would benefit from reporting a performance *spread* over a sufficient sample of plausible formats, instead of simply reporting the formatting used and its performance, as is currently standard. Moreover, we argue that this reporting is crucial when comparing the performance of different models, as we show the influence of formatting choices only weakly correlates between models, thus making and fixing a formatting choice could introduce a significant confounding factor.

Fully exploring the space of prompt formats is intractable, as computation costs scale linearly with the number of formats considered. FormatSpread efficiently explores the space of prompt formats under a user-specified computational budget using Bayesian optimization. FormatSpread does not require access to the model weights, allowing its use on API-gated models: we find a spread up to 56 accuracy points with a median spread of 6.4 accuracy points with GPT3.5 across 320 formats and 53 tasks at a cost of under 10USD on average per task. Beyond facilitating evaluation, we also propose a suite of analyses to

further characterize model sensitivity to formatting. Among other results, we show that the separability of continuous prompt embeddings correlates with the spread observed in task performance.

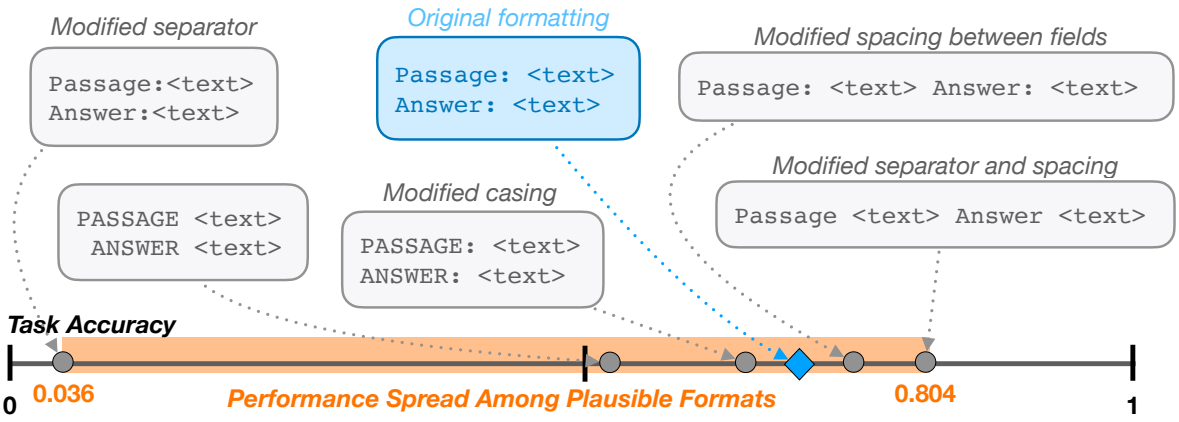


Figure 2.1: Slight modifications in prompt format templating may lead to significantly different model performance for a given task. Each `<text>` represents a different variable-length placeholder to be replaced with actual data samples. Example shown corresponds to 1-shot LLaMA-2-7B performances for task280 from SuperNaturalInstructions (Y. Wang et al., 2022a). This StereoSet-inspired task (Nadeem, Bethke, and Reddy, 2021) requires the model to, given a short passage, classify it into one of four types of stereotype or anti-stereotype (gender, profession, race, and religion).

2.2 Overview

We evaluate LLM performance over the space of prompt formats that may plausibly be chosen by a non-adversarial user when designing a prompt for a target task, where the space of formats is defined by a grammar (§2.3.1). Our grammar's definition naturally induces a definition of semantic equivalence among formats. We quantify model sensitivity in terms of performance range in a target task across the space of equivalent prompt formats to the original choice (§2.4.2). We cast the problem of searching across this space as a bandit problem, and propose FormatSpread (§2.3), which consists of a grammar (§2.3.1) and a procedure to estimate the minimum and maximum performance across a set of semantically equivalent formats given a pre-defined metric (§2.3.2). FormatSpread uses Bayesian optimization to identify the expected performance range with low additional computational cost (§2.4.5) all without requiring access to model weights, which enables use on API-gated LLMs. Furthermore, we perform in-depth analysis of this observed sensitivity, including by quantifying the contribution of individual feature choices to the final performance (§2.4.3) and measuring the identifiability of a format based solely on a model's internal, continuous representation of any prompt via correlation with model performance (§2.4.4).

2.3 Measuring Sensitivity with FormatSpread

2.3.1 Grammar of Plausible Prompt Formats

We construct a grammar that defines both the space of plausible prompt formats and semantic equivalence between formats. The grammar is manually constructed, as opposed to automatically induced from data, to guarantee a higher level of precision when defining the set of equivalent formats. Our grammar is directly tested by verifying that it can generate the formatting associated with 100+ Super-NaturalInstructions tasks (Y. Wang et al., 2022a).

Our grammar consists of fields that are composed to create a prompt format. For example, the format `'Passage: <text> || Answer: <text>'`, has basic fields `'Passage: <text>'`, and `'Answer: <text>'`, denoted a_1, and a_2. Each basic field consists of a *descriptor* (e.g. `'Passage'`), a *separator* (e.g. `': '`), and a text placeholder to replace with each data point. We define basic fields as $B_1(d, s, f) := f(d)s$`<text>` using Backus-Naur notation, where d is a descriptor string, $s \in \mathcal{S}_1$ a separator, and $f \in \mathcal{F}_{\text{casing}}$ a function that alters d while preserving meaning. Thus, in our example, $a_1 = B_1($`Passage`$, $`': '`$, id)$ and $a_2 = B_1($`Answer`$, $`': '`$, id)$, with id the identity function. We define joining several fields as $B_2^{(n)}(X_1, \ldots, X_n, c) := X_1 c X_2 c \ldots c X_n$, with $c \in \mathcal{C}$ being a *space*. Our example's prompt format may be written as $B_2^{(2)}(a_1, a_2,$ `' || '`$)$.

The grammar also supports enumeration, which is defined as joining several basic fields, each representing a different list item. For example, the enumeration `'Option (A): <text>, Option (B): <text>, Option (C): <text>'` may be written as $B_2^{(3)}(a_1, a_2, a_3,$ `' || '`$)$, where $a_i = B_1(e_i,$ `': '`$, id)$. In our example, e_1 represents `'Option (A)'`, and may in turn be written as the concatenation $e_i := d s_2 f_{\text{item}}(i)$ with $d =$ `'Option'`, $s_2 =$ `' '` (single space), and $f_{\text{item}}(1) =$ `'(A)'`. Each f_{item} transforms an item i using a number format (e.g. letters or Roman numerals, denoted as $\mathcal{F}_{\text{item2}}$) and an item wrapper (e.g. `(A)` or `[A]`, denoted as $\mathcal{F}_{\text{item1}}$).

In summary, we define valid prompt formats as those accepted by the following grammar:

$$
\begin{aligned}
B_0() &:= \texttt{<text>} \\
B_0'(d, s) &:= f(d)s \quad \text{with } s \in \mathcal{S}_1,\ f \in \mathcal{F}_{\text{casing}} \\
B_1(d, s, f) &:= f(d)s\texttt{<text>} \quad \text{with } s \in \mathcal{S}_1,\ f \in \mathcal{F}_{\text{casing}} \\
B_2^{(n)}(X_1, \ldots, X_n, c) &:= X_1 c \ldots c X_n \quad \text{with } c \in \mathcal{C},\ X_i \in \{B_0, B_0', B_1, B_2, B_3\}\ \forall i \\
B_3^{(n)}(d, j_1, \ldots, j_n, s_1, s_2, c, f_1, f_2) &:= B_2^{(n)}(B_1(e_1, s_1, f_2)), \ldots, B_1(e_n, s_1, f_2), c) \\
&\qquad \text{where } e_i := f_2(d)\ s_2\ f_1(j_i),\ j_i \in \mathbb{N}_0\ \forall i, \\
&\qquad s_1 \in \mathcal{S}_1,\ s_2 \in \mathcal{S}_2, f_1 \in \mathcal{F}_{\text{item}}, f_2 \in \mathcal{F}_{\text{casing}}
\end{aligned}
$$

Our grammar defines valid formats as finite compositions of B_0, B_0', B_1, B_2, B_3. The sets $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{C}$, $\mathcal{F}_{\text{casing}}$, $\mathcal{F}_{\text{item}}$ (two sets of separators, spaces, casing functions, and itemizing functions respectively) are pre-defined by the user. Throughout this work, we instantiate all sets with values typically observed in human-written prompt formats. We intentionally only modify the casing of descriptors (via $\mathcal{F}_{\text{casing}}$) to guarantee semantic equivalence; one may also define a set of functions that paraphrases the descriptor, e.g., via synonym replacement. Appendix A.1.2 contains the full list of values we use for the constant sets, as well as a visualization of a prompt template generated from the grammar.

Prompt Format Equivalence. Two prompt formats p_1, p_2 are equivalent if they represent the same rule application B_i, the descriptors (if any) are the same, and the sub-elements (if any) are equivalent. Appendix A.1.1 contains the formal definition of equivalence. The grammar's strict definition allows us to assume that sets of equivalent formats share equivalent meanings. When measuring sensitivity (§2.3.2), we explore only the space of formats equivalent to a task's original format.

Contextual Restrictions. We define restrictions to the combinations of spaces and separators to further ensure naturalness. For example, if $B_2(X_1,\ldots,X_n,c)$ where c does not contain a newline, then each X_i's separators and any subcomponents' separators should not contain a newline. This avoids unnatural formats like `Input:\n <text> Output:\n <text>`. We also allow for adding conditions that force constants (separators, spaces, etc.) in different applications of B_i to be equal. When measuring sensitivity to format perturbations, if two separators or spaces are equal in an original format, they are forced to jointly change to be considered equivalent. Appendix A.1.3 contains all contextual restrictions.

Final Prompt Construction. Given a valid format p accepted by the grammar, the final prompt is constructed by concatenating with space c an instruction string *inst*, n few-shot data points $D_1, \ldots, D_n$ exemplifying the task, and a data point D_{n+1} to be solved. All few-shot examples D_i are formatted using p. Thus, the final prompt template is: $inst\ c\ p(D_1)\ c\ p(D_2)\ c\ \ldots\ c\ p(D_n)\ c\ p(D_{n+1})$. Since D_{n+1}'s output will be generated by the model, an empty string is added in place of the answer in the last field in the template. Prompt construction will modify *inst* to match specific choices encoded in p: concretely, if p enumerates valid multiple-choice options as characters $x_1 \ldots x_n$, we ensure *inst* refers to these choices as $x_1 \ldots x_n$.

2.3.2 Measuring Sensitivity

We measure how plausible choices in prompt formatting influence quantifiable metrics of generated outputs. Given a set of plausible formats $\{p_1, \ldots, p_n\}$, a dataset $\mathcal{D}$, and a scalar metric m, let the *performance interval* be $[\min_i m(p_i, \mathcal{D}), \max_i m(p_i, \mathcal{D})]$. We define the *performance spread* or simply *spread* as $\max_i m(p_i, \mathcal{D}) - \min_i m(p_i, \mathcal{D})$. Higher spread indicates more sensitivity to variance within the space of plausible, semantically-equivalent formats. While our method is agnostic to the scalar metric m used, and one could consider a number of metrics including text length, formality, or toxicity, throughout this work we focus our analysis on estimated task accuracy *acc*. Due to ease in automatic evaluation, here we evaluate on classification tasks.

Our goal is to compute spread for a given model and task. A comprehensive approach would be to fully evaluate each plausible format p_i on the entire evaluation dataset $\mathcal{D}$. This increases the cost of reporting a model's performance linearly with n, which becomes computationally infeasible for large values of n. Following prior gradient-free prompt engineering work (Pryzant et al., 2023; Y. Zhou et al., 2023), we model our problem as a multi-arm bandit. Given a random sample of n formats (arms) $p_1, \ldots, p_n$ for a task, an arm p_i's hidden value is the actual performance $m(p_i, \mathcal{D})$ when evaluated on the full dataset $\mathcal{D}$, and the

reward for pulling the arm is an estimate $m(p_i, \tilde{\mathcal{D}})$ where $\tilde{\mathcal{D}} \subset \mathcal{D}$, $|\tilde{\mathcal{D}}| = B$ (mini-batch size) and no element of $\tilde{\mathcal{D}}$ has yet been evaluated with p_i.

We assume a budget of E total data point evaluations. We first search for the highest performing format with budget $E/2$, and then for the lowest performing format with budget $E/2$. Evaluations done for the first exploration are readily available for the second exploration, which yields a more informative prior for many formats. We consider two well-known regret minimization bandit algorithms: Thompson sampling (used in FormatSpread) and Upper Confidence Bound (UCB).

Thompson Sampling. This simple, high-performing Bayesian inference heuristic randomly draws each arm according to its probability of being optimal (Chapelle and L. Li, 2011). Each $m(p_i, \mathcal{D})$ is modeled as a random variable, and since with our target metric each data point evaluation is a Bernoulli trial, it is natural to model $m(p_i, \mathcal{D})$ as a Beta distribution. In each round, Thompson sampling draws from each $m(p_i, \tilde{\mathcal{D}})$ and chooses the best arm $\hat{i}$ (Algorithm 1). It then updates $\hat{i}$ according to the number of observed successes r, and the corresponding $B - r$ failures, within $\tilde{\mathcal{D}}$.

Algorithm 1 Thompson Sampling for Bernoulli Bandits

$S_i^{(1)} \leftarrow 0, N_i^{(1)} \leftarrow 0$ (success counters and total times armed was drawn counter)
for $t \leftarrow 1, \dots E/B$ **do**
  for $i \leftarrow 1, \dots, K$ **do**
    Take $\theta_i^{(t)}$ from Beta$(\alpha_i + S_i^{(t)}, \beta_i + (N_i^{(t)} - S_i^{(t)}))$
  Draw arm $\hat{i} = \arg\max_i \theta_i^{(t)}$ (or $\arg\min$ in minimization problems) and observe reward r
  $S_{\hat{i}}^{(t+1)} \leftarrow S_{\hat{i}}^{(t)} + r, \quad N_{\hat{i}}^{(t+1)} \leftarrow N_{\hat{i}}^{(t)} + B$

Thompson sampling allows for setting informative priors (α_i, β_i) based on domain knowledge to accelerate runtime. Appendix A.1.4 details the exact priors we use. To our knowledge, we are the first to consider a Bayesian sampling method for prompt optimization.

Upper Confidence Bound (UCB) Sampling. UCB (Lai, Robbins, et al., 1985) computes an upper confidence bound to each arm's performance, derived from Chernoff's bound. The key difference with Thompson sampling is in how $\theta_i^{(t)}$ is defined. In UCB's frequentist approach, $\theta_i^{(t)}$ is assigned the estimated accuracy plus the upper confidence bound: $\theta_i^{(t)} \leftarrow S_i/N_i + c\sqrt{log(t)/N_i}$. We use $c = 2$ following Pryzant et al., 2023, who find UCB with $c = 2$ to be most effective for prompt optimization.

Naive Sampling. Each prompt format is evaluated on E/n points (with appropriate rounding).

2.4 Characterizing Prompt Format Variance with FormatSpread

2.4.1 Experimental setup

Data. We use a subset of 53 tasks from Super-NaturalInstructions (Y. Wang et al., 2022a) with diverse human-written formats and instructions, comprising 19 multiple-choice tasks and 34 classification tasks with $\{2, 3, 4\}$ basic fields. Appendix A.2.1 details the exact task selection procedure. To construct the final prompt template, we concatenate each task's instruction and n formatted few-shot examples using `\n\n` as spacing. While selection and ordering of few-shot examples is a component of prompt design influencing features of model output (Lu et al., 2022), our work focuses on prompt formatting. To remove this confounder, we fix the exact choice and ordering of examples for each task and for a given number of shots n. Few-shot examples for each task are chosen randomly within each dataset and are not used for evaluation. We evaluate task data samples on an arbitrary order fixed across settings. Datasets are assumed to be of size 1,000 for fair evaluation across tasks.

Models. We evaluate LLaMA-2-{7B,13B,70B} (Touvron et al., 2023a), Falcon-7B and Falcon-7B-Instruct (Almazrouei et al., 2023), GPT-3.5-Turbo (OpenAI, 2022), all autoregressive LMs.

Task Evaluation Metrics. We use two popular measures for computing accuracy: exact prefix matching and probability ranking. In exact prefix matching, we check if the output's prefix matches the expected answer after normalization (casing, spacing, newlines). Ranking accuracy computes the rate that the expected answer is the highest-ranked valid option (in multiple choice and classification tasks) according to the model's output distribution. Results are reported using ranking accuracy unless specified otherwise. Appendix A.2.2 shows additional analysis of exact prefix matching, with spreads even higher than those shown in Section 2.4.2, and including how formatting choice affects task degeneration (i.e., not answering any valid option).

2.4.2 Prompt formats have a large performance spread, not eliminated by increasing few-shot examples or model size, nor with instruction tuning

For each evaluation task we randomly sample 10 plausible prompt formats and use FormatSpread to compute performance spread for each modeling and n-shot choice (Figure 2.6). We find significant performance spread across tasks, with a median spread of 7.5 accuracy points across choices in the model and the number of few-shot examples. 20% of tasks consistently result in a spread of at least 15 accuracy points for all LLaMA-2 settings, and at least 9 points for all Falcon settings. We observe several tasks with performance spread over 70 accuracy points. Because this analysis uses only 10 randomly sampled formats, it represents a lower bound of the true spreads for each task. Furthermore, there exists significant performance spread regardless of increased model size (Figure 2.2

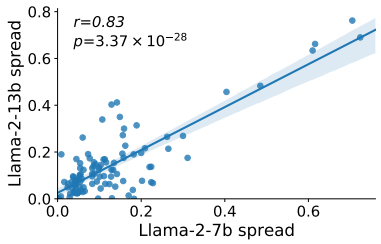


Figure 2.2: Llama-2-7B vs. Llama-2-13B

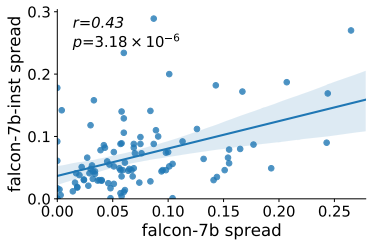


Figure 2.3: Falcon-7B vs. Falcon-7B-Instruct

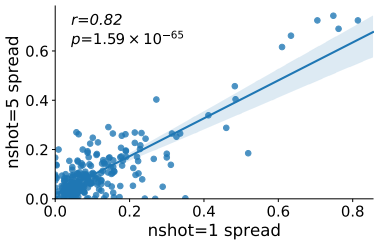


Figure 2.4: 1- vs. 5-shot (same task, model)

Figure 2.5: Spread comparison between evaluating the same task under different models or n-shots.

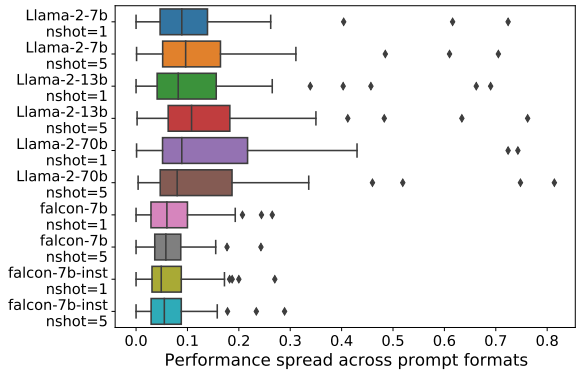


Figure 2.6: Spread across models and n-shots.

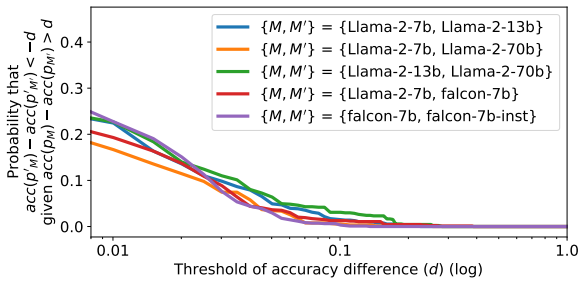


Figure 2.7: Probability that model M performs *worse* than M' by at least d when using format p', given that M performed *better* than M' by at least d using format p. 53 tasks, 1- and 5-shot.

and Figure A.2 for Llama-2-70B), instruction tuning (Figure 2.3), or number of few-shot examples (Figure 2.4; Figure 2.2 and 2.3 plot 1- and 5-shot jointly). Appendix A.2.2 demonstrates similar results on a selection of non-classification tasks, and expands the spread discussion to plotting the entire accuracy distribution, along with a dispersion metric.

Comparison trends between models are often reversed just by choosing different formats. Assuming model M is better than M' by at least d accuracy using prompt p, we compute how often M' achieves at least d higher accuracy than M under a different format p'. Figure 2.7 shows these trends are often reversed: LLaMA-2-13B and -70B reverse trend by at least $d = 0.02$ with probability 0.141; LLaMA-2-7B and Falcon-7B reverse trend by at least $d = 0.02$ with probability 0.140. Strikingly, often both experiments (first using p, and then p') were statistically significant (p-value < 0.05) on 1000 samples[1]: 76% and 47% respectively for the two model comparisons mentioned above. We find that formats yielding high performance for model M may not yield high performance for M', implying that **formats may not be inherently good or bad** (Appendix A.2.2).

[1]We use one-sided McNemar tests, also known as paired χ^2 tests, since we evaluate models on the same set of samples. We test the significance of M being *better* than M' under p, and M being *worse* than M' under p'.

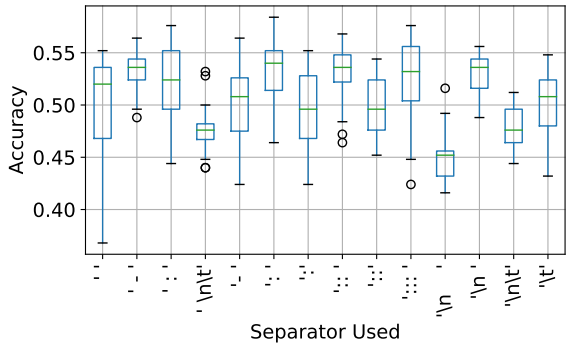


Figure 2.8: Example of accuracy variance for different choices of constants in $\mathcal{S}_1$ for task1283.

Table 2.1: Tasks where at least two constants yield different performance (*weakly* different if their boxes in a boxplot do not overlap, *strongly* if boxes including whiskers do not overlap).

	Median Spread (range [0, 1])	Weak Diff.	Strong Diff.
$\mathcal{C}$	0.144	29%	1%
$\mathcal{S}_1$	0.132	43%	22%
$\mathcal{S}_2$	0.238	0%	0%
$\mathcal{F}_{\text{item1}}$	0.176	0%	0%
$\mathcal{F}_{\text{item2}}$	0.173	45%	10%
$\mathcal{F}_{\text{casing}}$	0.188	3%	0%

2.4.3 How do individual features contribute to performance?

We analyze how choices in particular constants (i.e. $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{C}$, $\mathcal{F}_{\text{casing}}$, $\mathcal{F}_{\text{item}}$) independently influence task performance across different formats. Figure 2.8 shows the distribution of accuracy for 500 sampled prompts conditioned on the choice of $\mathcal{S}_1$ (the separator between a descriptor and the text placeholder) for one task in Super-NaturalInstructions. When comparing the individual influence of two feature choices, we measure both *weak* and *strong* notions of dissimilarity between distributions of accuracy across prompts conditioned on a chosen feature. We say two constant choices yield *weakly* different accuracy distributions if the values between the first quartile (Q_1) and third quartile (Q_3) do not intersect. This is equivalent to the boxes in a boxplot not overlapping. We say two constant choices yield *strongly* different accuracy distributions if the ranges $[2.5Q_1 - 1.5Q_3, 2.5Q_3 + 1.5Q_1]$ do not overlap (adjusted to end in a data point). This is equivalent to two boxplots with their whiskers not overlapping. In Figure 2.8, `' \n\t'` and `': '` (fourth and sixth) are only weakly different.

We compute accuracy for 500 random formats with 250 samples each on 31 tasks for 1-shot Llama-2-7B. Table 2.1 shows that choices in $\mathcal{S}_2$, $\mathcal{F}_{\text{item1}}$, $\mathcal{F}_{\text{casing}}$ do not independently predict performance differences (weakly or strongly): although these features can have a large performance variance and thus should be explored with FormatSpread, they cannot be used to independently predict accuracy changes. Other constant sets have varying degrees of differences, with $\mathcal{S}_1$ (separators) and $\mathcal{F}_{\text{item2}}$ (number format changes in enumerations) having the most individual impact. All tasks with strong dissimilarities are shown in Appendix A.2.4.

Small prompt variations often yield large performance differences. Table 2.2 shows a selection of tasks where changing a single constant on a format (e.g., casing in task322) results in large accuracy differences. Figure 2.9 shows that regardless of the scoring criterion used, a significant ratio of these atomic changes are associated with large accuracy changes. For example, 24% of atomic changes have an associated accuracy change of at least

Table 2.2: Examples of atomic changes' impact on accuracy using probability ranking (prefix matching shown in Table A.2). `{}` represents a text field; p_2 yields higher accuracy than p_1 for all tasks.

Task Id	Prompt Format 1 (p_1)	Prompt Format 2 (p_2)	Acc p_1	Acc p_2	Diff.
task280	`passage:{}\n answer:{}`	`passage {}\n answer {}`	0.043	0.826	0.783
task317	`Passage::{} Answer::{}`	`Passage:: {} Answer:: {}`	0.076	0.638	0.562
task190	`Sentence[I]- {}Sentence[II]- {} - Answer\t{}`	`Sentence[A]- {}Sentence[B]- {} - Answer\t{}`	0.360	0.614	0.254
task904	`input:: {} \n output:: {}`	`input::{} \n output::{}`	0.418	0.616	0.198
task320	`target - {} \n{} \nanswer - {}`	`target - {}; \n{}; \nanswer - {}`	0.361	0.476	0.115
task279	`Passage : {}. Answer : {}`	`PASSAGE : {}. ANSWER : {}`	0.372	0.441	0.069
task322	`COMMENT: {} ANSWER: {}`	`comment: {} answer: {}`	0.614	0.714	0.100

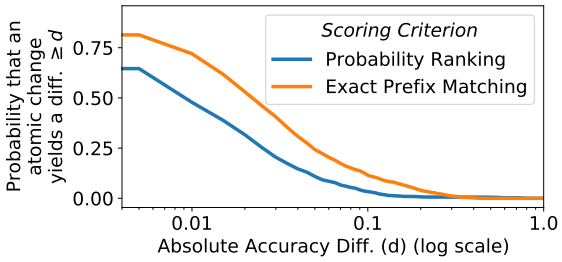


Figure 2.9: Probability that an atomic change (e.g. changing a space, separator) has a given impact in accuracy for two scoring criteria. 53 tasks, 30 sampled atomic changes each.

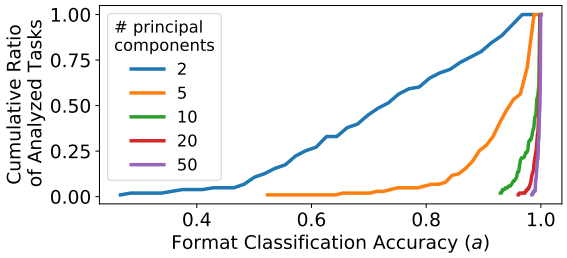


Figure 2.10: Cumulative ratio of tasks that can be classified with at most a accuracy using the top principal components of the last decoding layer of the prompt.

5 points when using exact prefix matching as scoring criteria (11% when using probability ranking).

The space of prompt format accuracy is highly non-monotonic, which makes local search algorithms over the space less effective. Let (p_1, p_2, p_3) be a prompt format triple such that p_{i+1} is obtained by making an atomic change to p_i. We argue that if the prompt format space is smooth, we should often see a triples' accuracy to be strictly monotonic over i. We choose 24 tasks (13 multiple choice, 11 non-multiple choice), sample 300 (p_1, p_2, p_3) triples for each, and the compute accuracy (using exact prefix matching) of each p_i on 250 samples. 32.4 and 33.6% of triples were monotonic for multiple-choice and non-multiple-choice tasks respectively. Given that random shuffling within a triple will result in monotonicity 33.3% of the time, this suggests that local search mechanisms like simulated annealing may not be effective as they require a locally smooth search space.

2.4.4 Prompt formats are identifiable transformations of prompt embeddings

Prompt format choices represent a deterministic transformation of the input, even if its impact on the resulting performance is hard to predict. We represent prompt embeddings

as the last hidden layer obtained when processing the whole input prompt (immediately before generating the first token). We demonstrate that format choice yields a highly identifiable transformation over this embedding, which suggests that formats can be seen as transformations of the output probability distribution.

For each task, and for both {1, 5}-shot settings, we collect prompt embeddings from LLaMA-2-7B corresponding to 10 randomly sampled valid formats for 1000 evaluation examples. We train an XGBoost (T. Chen and Guestrin, 2016) classifier that maps from the top n principal components of a prompt embedding to the prompt format.[2] We find that although the original prompt embeddings are of size 4,096[3], using just the top 100 principal components can result in a classifier with $\geq$0.98 accuracy in format identification for all 31 tasks analyzed. Figure 2.10 shows the accuracy of format classification given a fixed number of principal components.[4] We find that classifier accuracy given just the top two components correlates moderately with the spread of performance in the prompts they represent (0.424, $p = 8.04 \cdot 10^{-6}$; 0.555 for the 5-shot setting; using exact prefix matching).

2.4.5 Fast exploration of the prompt formatting space: FormatSpread

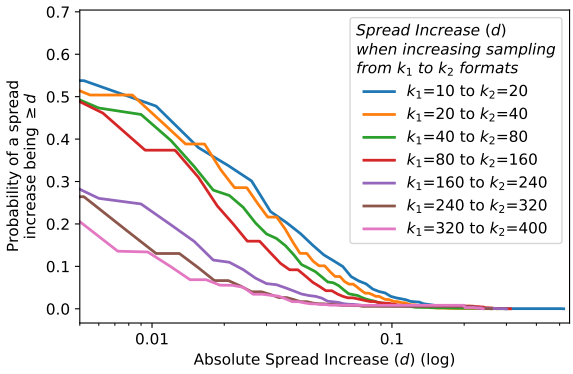


Figure 2.11: Probability of observing a spread increase of at least d when increasing sample size from k_1 to k_2 formats. 31 tasks, 100 trials each.

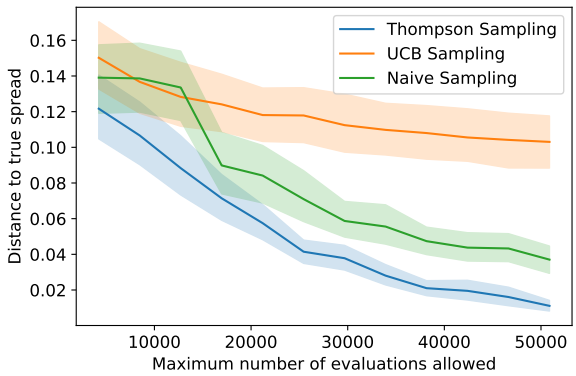


Figure 2.12: Difference between the true sample spread and each algorithm-found spread with respect to E (evaluation budget). 320 formats, $B = 20$, average of 5 trials over 31 tasks shown.

In Section 2.4.2, we demonstrate that even when sampling just 10 formats from the space of plausible formats, we still observe significant performance spread on many tasks. However, this is only a lower bound of the spread a task may exhibit when increasing the number of formats: for example, about 17% of tasks are expected to increase their spread by at least 5 accuracy points when increasing from 10 to 20 sampled formats. Figure 2.11 quantifies

[2]We train with 800 vectors from each of the 10 formats (8000 vectors) and evaluate on the remaining 200.

[3]Equivalent to the dimension of hidden representations for LLaMA-2-7B.

[4]Figure A.24 in the Appendix visualizes examples of the top two principal components for ten prompt formats.

the expected increase in spread when increasing the number of formats by evaluating 500 formats on 250 samples each and computing expected gains.

Figure 2.12 compares the efficiency of Thompson sampling, UCB, and naive sampling for estimating spread with respect to a budget E (Section 2.3.2). To ensure accurate reports, we compute and show the true spread of the highest- and lowest-performing formats chosen by each method using all data. With a budget of 51,200 evaluations, Thompson sampling results in a spread within 1 accuracy point of the true spread, while naive sampling finds a spread within 4 points, and UCB within 11.

Finally, we use FormatSpread to measure sensitivity of several models where inference is expensive. With a budget of 40,000 evaluations and 320 prompt formats, we find that 1-shot LLaMA-2-70B–ran using 4-bit quantization (Dettmers et al., 2022)–yields a median spread of 0.171 (mean=0.221, std=0.200, using probability ranking across 53 tasks; 25% of tasks had a spread of 0.292 or higher, with a maximum spread of 0.876), and GPT-3.5 yields a median spread of 0.064 (mean=0.110, std=0.115, across 53 tasks using exact prefix matching given that we do not have access to the full logits; 25% of tasks had a spread of 0.148 or higher, with a maximum spread of 0.562), showing sensitivity to formatting is still present even on larger models. 5-shot LLaMA-2-70B still shows high spreads, with 25% of tasks having a spread of 0.310 and a maximum of 0.841. See spread visualization in Figure A.28, and a list of best and worst formats found in Table A.4.

2.5 Related Work

The task of automatically finding the best-performing prompt for a given task without changing model parameters has recently gained attention, given the constantly improving yet somewhat unpredictable performance of LLMs. Prior work has often focused on discovering optimal prompts with gradient-based methods, which are effective, but often lead to disfluent or unnatural prompts (Shin et al., 2020), which can be mitigated with a Langevin dynamics-based method (W. Shi et al., 2023). Another approach is to learn, optimize, and insert continuous representations of prompts and tasks as input to models (Ding et al., 2022; Ilharco et al., 2023; Lester, Al-Rfou, and Constant, 2021; G. Qin and Eisner, 2021). These methods also require access to the LLM's parameters, thus cannot be applied to models behind an API. In contrast, FormatSpread does not assume access to any model internals. Prior gradient-free work has focused on edit-based enumeration over human-written prompts (Prasad et al., 2023), reinforcement learning (Deng et al., 2022), and by using LLMs themselves (Gao, Fisch, and D. Chen, 2021; Y. Zhou et al., 2023). These works aim to achieve competitive task performance, even if the meaning of the prompt or instruction is modified. To our knowledge, we are the first to focus specifically on prompt formatting variance, a quintessential example of semantic equivalence.

Jailbreaking refers to the behavior of intentionally manipulating prompts to elicit inappropriate or sensitive responses, or otherwise reveal parts of the prompt that were intentionally not revealed. While the objective differs from our work, jailbreaking works (A. Wei, Haghtalab, and Steinhardt, 2023; Zou et al., 2023) share the underlying technical question of finding the lowest-performing prompt. Our methods differ, since A. Wei, Haghtalab, and Steinhardt, 2023 evaluate human-generated attacks to guide adversarial prompt design, and

Zou et al., 2023 uses gradient-based search methods simultaneously across multiple models.

Some existing work has explored the influence of certain prompt design choices on model performance, for example the prompt's language (Gonen et al., 2023), the ordering of few-shot examples (Lu et al., 2022), and their patterns (Madaan, Hermann, and Yazdanbakhsh, 2023). Other work has focused on providing textual interpretations of continuous prompt representations (Khashabi et al., 2022). Beyond autoregressive LLMs, existing work has focused on performance variance in masked language models (Elazar et al., 2021; Z. Jiang et al., 2020). Our work follows efforts in other domains that explore the influence of spurious features on research evaluations, e.g., in deep reinforcement learning (Henderson et al., 2018; Islam et al., 2017) and statistical machine translation (J. H. Clark et al., 2011).

2.6 Discussion

We introduce FormatSpread, an algorithm that estimates the performance *spread* across prompt formatting choices.[5] We use FormatSpread to evaluate the spread of several widely-used open-source LLMs for classification tasks in few-shot learning settings. We find that spread is large regardless of model choice, even when increasing model size, number of few-shots, or when using instruction tuning. FormatSpread is designed to efficiently search the space of plausible prompt formats under a user-specified computational budget. For example, with a computational budget of exploring only 5% of the entire search space for task with 2,500 test examples and 320 plausible formats, we are able to estimate spread within 2 accuracy points of the true spread.

We also characterize the space of prompt formats, finding that it is largely non-monotonic and that few atomic features can be predictors of performance alone, although the separability of format embeddings is highly correlated with observed performance spread. These findings informed the design of our search procedure, where local search methods are not advantageous.

Our findings suggest that performance spread caused by arbitrary prompt formatting choices may influence conclusions made about model performance, especially when comparing models on benchmark tasks. Thus, we recommend that work evaluating LLMs with prompting-based methods would benefit from reporting a range of performance across plausible formats. However, we want to emphasize that single-format evaluation may still be sufficient for many use cases. For example, for researchers or practitioners who build systems on top of LLMs, choosing a single prompt format that works sufficiently well for use in this larger system is a valid methodological choice. However, we encourage future research to compute FormatSpread when comparing their systems to out-of-the-box models, to ensure fair baseline representation. Furthermore, FormatSpread can be used to identify lower-bound performance of a model or system. For example, when using a model for socially impactful tasks, such as stereotype classification in Figure 2.1, it is important to report the range of accuracy a non-adversarial user might encounter. Likewise, it is crucial to consider robustness to spurious features when claiming that models possess general abilities, such as theory of mind; and beneficial to report when e.g. exploring model biases. We leave

[5] We thoroughly describe the limitations of our method in Section 2.6.

it to future research to develop regularization procedures either during training or with an already-trained model to make models robust to diverse formatting choices.

Limitations

As defined by our grammar, all equivalent formats are semantically equivalent to human readers. However, some of them are more likely to be used by humans than others. Spaces and separators are inspired from naturally-occurring formats, but some values are more unusual, such as the spacing `<sep>` or the separator `::`. Contextual restrictions enable disallowing undesired combinations of e.g. spaces and separators. However, formats may have multiple valid parses, and some may be more prone than others to unnatural character combinations. For example, let a data sample be `'Passage: Lorem ipsum dolor sit amet. Answer: Yes'`. Depending on if we consider the full stop `.` to be part of the passage or the format, we may parse it as $B_2^{(2)}(B_1(\texttt{Passage}, \texttt{': '}, id), B_1(\texttt{Answer}, \texttt{': '}, id), \texttt{' '})$ or $B_2^{(2)}(B_1(\texttt{Passage}, \texttt{': '}, id), B_1(\texttt{Answer}, \texttt{': '}, id), \texttt{'. '})$. In this work, we choose the former parsing throughout tasks to ensure full sentences. This sometimes[6] leads equivalent formats to have a less usual, yet trivially semantically equivalent resulting character combinations, e.g. $B_2^{(2)}(B_1(\texttt{Passage}, \texttt{': '}, id), B_1(\texttt{Answer}, \texttt{': '}, id), \texttt{'; '})$. This last format would have the following string form on the example above: `'Passage: Lorem ipsum dolor sit amet.; Answer: Yes'`. We observe high performance spread both in these cases and beyond them. Contextual relations may also restrict these cases if desired by the end user.

Additionally, we focus our evaluation on tasks that have reasonably short input instructions and input field length (see task selection details in Appendix A.2.1). Future work may investigate on how input length affects final performance.

2.7 A 2026 Analysis on the Influence of Tokenization and Reasoning Models

FormatSpread on tokenizer-free architectures The findings shown during the chapter focused on models trained with BPE tokenization, prompting the question of whether format sensitivity is an artifact of how BPE carves text into subword units rather than a deeper property of transformer-based language models. BLT (Pagnoni et al., 2025) offers a direct test, as a 7B-parameter model bypassing fixed-vocabulary tokenization entirely and instead operating at the byte level and letting the model learn its own dynamic grouping of bytes. BLT was released nine months after the study shown in this chapter.

We ran FormatSpread on BLT-7B across 53 SuperNaturalInstructions tasks (the same task set as the original evaluation) with the same number of formats and evaluation budget. The results, shown in Figure 2.13, reveal substantial spread that remains regardless of the number of few-shot examples. These results suggest that tokenization may not be the sole driver of format sensitivity.

[6]Less than 20% of cases, based on a manual inspection of 10 formats across 20 tasks.

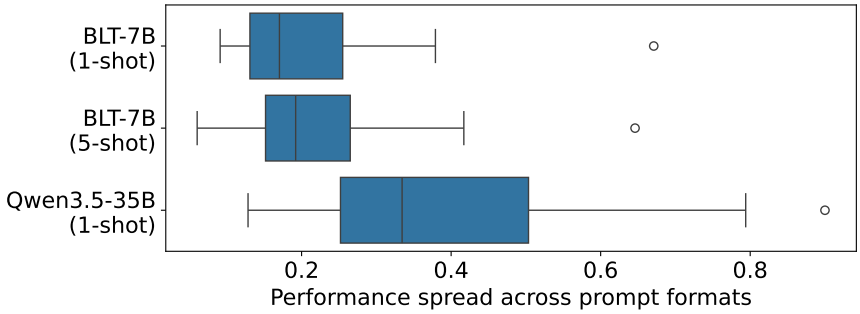


Figure 2.13: Performance spread across prompt formats for BLT-7B (1-shot and 5-shot) and Qwen3.5 (1-shot, no chain-of-thought) across SuperNaturalInstructions tasks, evaluated using probability ranking. Qwen3.5 35B, a much larger and more capable 2025 model, exhibits even greater spread, showing that scale and recency alone do not resolve format sensitivity.

On format sensitivity in 2026 and beyond Figure 2.13 also includes Qwen3.5-35B (released in February 2026) evaluated without thinking mode using probability ranking—the same evaluation protocol as the original paper. Despite being a much larger and more recent model, Qwen3.5-35B exhibits *greater* spread than BLT-7B across the same tasks, consistent with the original finding that scale alone does not resolve format sensitivity. We additionally ran the full FORMATSPREAD algorithm (320 formats, 100 samples per format, Thompson sampling) with Qwen3.5-397B on five representative tasks via the Together AI API.[7] Even at mean accuracies of 87–96%, raw spreads reach 15 to 46 accuracy points. A notable share is driven by outputs that could not be parsed under exact prefix matching; normalizing these out leaves near-zero genuine capability spread on some tasks (e.g., task214: 0 points) but substantial real spread on others (task297: 37 points, task580: 11 points).

As a lower bound, we also ran a small comparison on fully evaluating 10 formats across the same five tasks with Qwen3.5-397B—the largest size available for Qwen3.5. Even in this minimal setup, non-trivial spread remains (task580: 8 points, task1297: 6 points), suggesting that thinking mode reduces but does not eliminate format sensitivity.

More broadly, as tasks become easier for newer models we expect the absolute magnitude of spread to decline; we leave a comprehensive re-evaluation on more recent and challenging tasks to future work. Nonetheless, given recent benchmarking studies documenting persistent prompt sensitivity (Romanou et al., 2026), we expect similar behaviors to continue to appear,

[7]Because the Together AI API does not expose log-probabilities, these runs use exact prefix matching rather than probability ranking. All values in percentage points; "normalized spread" removes outputs that could not be parsed before computing the best-minus-worst gap.

Task	Mean acc.	Raw spread	Norm. spread
task214	96	46	0
task1297	96	18	6
task220	94	15	3
task580	87	38	11
task297	11	44	37

particularly for challenging tasks and smaller models.

Part II

Modeling Semantic Structure for Compositional Reasoning

Chapter 3

Limits of Transformers on Compositionality

*This chapter is based on **Faith and Fate**: Limits of Transformers on Compositionality. Published at NeurIPS 2023 (Spotlight).*

Chapter 2 showed how surface-level structure enabled exposing large, systematic variance in model performance that would otherwise go undetected. However, surface structure leaves the *content* of the task unexamined.

This chapter leverages semantic-level structure in the form of computation graphs. By recasting a task as a computation graph, what would otherwise be an unstructured input-output evaluation becomes a structured object amenable to complexity analysis, step-level verification, and sub-graph memorization analysis, operationalizing the thesis's core argument at the semantic level. An intentional methodological choice is to focus on algorithmic tasks where the solution intrinsically depends on solving similar sub-problems, and requires relatively few distinct operation types (which may have otherwise forced the inclusion of distinct edge types in the computation graph). This eases both the structural modeling and the quantification of the correlation between task successes and having been trained on the solutions to relevant sub-structures during training.

The findings point to principled limits on compositional generalization: performance that appears strong under low complexity degrades predictably as task structure grows, consistent with our theoretical predictions of error propagation under autoregressive generation. Part III will later ask whether structure-guided approaches extend to a key domain requiring social rather than purely mathematical reasoning, one where models face not only the compositional pressures documented here, but also the additional challenge of tracking the shifting mental states of multiple agents, and whether they can do more than diagnose failures.

3.1 Introduction

Large-scale transformers such as ChatGPT (OpenAI, 2022) and GPT4 (OpenAI, 2023) demonstrate unprecedented capabilities (Brown et al., 2020; Choi et al., 2023; OpenAI, 2022; Thoppilan et al., 2022; Zheng and Zhan, 2023), even noted as "sparks of AGI" (Bubeck et al., 2023). In stark contrast, the same models sometimes struggle with simple, intuitive tasks (Bian et al., 2024; Koralus and Wang-Mascianica, 2023; C. Qin et al., 2023). For instance, humans can solve 3-digit by 3-digit multiplication arithmetic after learning basic calculation rules (Dehaene et al., 2004; Hiebert, 2013). Yet, off-the-shelf ChatGPT and GPT4 achieve only 55% and 59% accuracies on this task, respectively (§3.3).

The striking discrepancy between the impressive successes of transformer LLMs on *seemingly complex* tasks and the astonishing failures on *seemingly trivial* tasks spark critical open questions about how to faithfully interpret their mixed capabilities. Under what conditions do transformers succeed, fail, and why? What types of errors do they make? Can transformers uncover implicit problem-solving rules or be taught to follow reasoning paths?

Seeking thorough answers to these questions remains an open research challenge. However, we offer novel insights into the fundamental limits of transformers[1], centered around *compositional problems* that require strict multi-hop reasoning to derive correct predictions. Applying step-by-step reasoning is fundamental to human intelligence (Simon and Newell, 1971; Simon, 1962). These compositional problems present compelling challenges for AI systems as they require combining basic reasoning operations to follow computational paths that arrive at unique correct solutions. In particular, we study two representative compositional tasks: long-form multiplication and a classic dynamic programming problem.

We propose two hypotheses. **First**, transformers solve compositional tasks by reducing multi-step compositional reasoning into linearized path matching. This contrasts with the systematic multi-step reasoning approach that learns to apply underlying *computational rules* required for building correct answers (Evans, 1989; Johnson-Laird, Khemlani, and Goodwin, 2015; Stenning and Van Lambalgen, 2012). Shortcut learning (Geirhos et al., 2020) via pattern-matching may yield fast correct answers when similar compositional patterns are available during training but does not allow for robust generalization to uncommon or complex examples. **Second**, due to error propagation, transformers may have inherent limitations on solving high-complexity compositional tasks that exhibit novel patterns. Errors in the early stages of the computational process can lead to substantial compounding errors in subsequent steps, preventing models from finding correct solutions.

To investigate our hypotheses, we formulate compositional tasks as *computation graphs*. These graphs break down problem-solving into submodular functional steps, enabling structured measurements of complexity and verbalization of computational steps as input sequences to language models. Moreover, we leverage information gain to predict patterns that models are likely to learn based on the underlying task distribution without the need to perform full computations within the graph.

Empirical results show that training on task-specific data leads to near-perfect performance on in-domain instances and under low compositional complexity, but fails drastically

[1]For brevity, we use 'transformers' to refer to 'autoregressive transformer LLMs' throughout the paper.

on instances outside of this region. This substantial gap suggests that systematic problem-solving capabilities do not emerge from maximum likelihood training (Bengio, Ducharme, and Vincent, 2000) on input-output sequences, even when prompted or trained with human-like reasoning steps (i.e., a linearization of computation graphs; §3.3.1). Models' success can be attributed, in part, to their exposure to training examples sub-graphs that involve the same computations required for solving test examples (see Section 3.3.2.2) Importantly, we provide theoretical evidence of exponential error accumulation using abstract compositional tasks. All tasks analyzed empirically in this paper are instantiations of these abstractions (§3.4). We argue that transformers could be inherently limited in solving compositionally complex tasks out-of-the-box.

As transformers continue to make tangible real-world impacts, it is pressing to interpret their remarkable performance critically. Our work takes a realistic look at the limitations of transformers in the context of compositional tasks. To shed light on practical future steps, we identify directions for addressing these limitations, such as using transformers for tasks that could be decomposed into few reasoning steps, tasks where evaluation may afford some leniency, and using transformers in combination with planning modules or refinement methods to improve their generations. To advance language AI, fundamental innovations are required to address or complement these limitations.

3.2 Measuring Limitations of Transformers in Compositional Tasks

Human problem-solving skills can be conceptualized as a graph structure, where each vertex represents a partial solution and the edges represent operators that can be applied to modify these solutions. As we will outline next and illustrate in Figure 3.1, we use computation graphs and corresponding metrics to methodically evaluate transformers' reasoning abilities.

3.2.1 Computation Graph Definition

Let A be a deterministic algorithm (function), and let $\mathcal{F}_A$ be a set of primitives (functions) the algorithm uses in its execution. Assuming the inputs $\mathbf{x}$ to algorithm A are given, we define $A(\mathbf{x})$'s static computation graph $G_{A(\mathbf{x})}$. $G_{A(\mathbf{x})} = (V, E, s, op)$ is a directed acyclic graph. Nodes V represent all variables' values during A's execution: each node $v \in V$ has a value $s(v) \in \mathbb{R}$ associated. Edges E represent the function arguments involved in some computation: for each non-source node $v \in V$, let $U = \{u_1, \ldots, u_j\} \subset V^j$ be its parent nodes. Then, $s(v) = f(u_1, \ldots, u_j)$ for some $f \in \mathcal{F}_A$. Since each node v is uniquely defined by the computation of a single primitive f, we define $op : V \to \mathcal{F}_A$ as $op(v) = f$.

Let $S \subset V$ be the source nodes of $G_{A(\mathbf{x})}$ and without loss of generality, let $o \in V$ be its sole leaf node. By definition, $S \equiv \mathbf{x}$ and $A(\mathbf{x}) = s(o)$, representing the input and output of A respectively.

To be able to train and evaluate a language model's ability to follow algorithm A we must linearize $G_{A(\mathbf{x})}$. Since we only consider autoregressive models, this linearization must

also be a topological ordering.

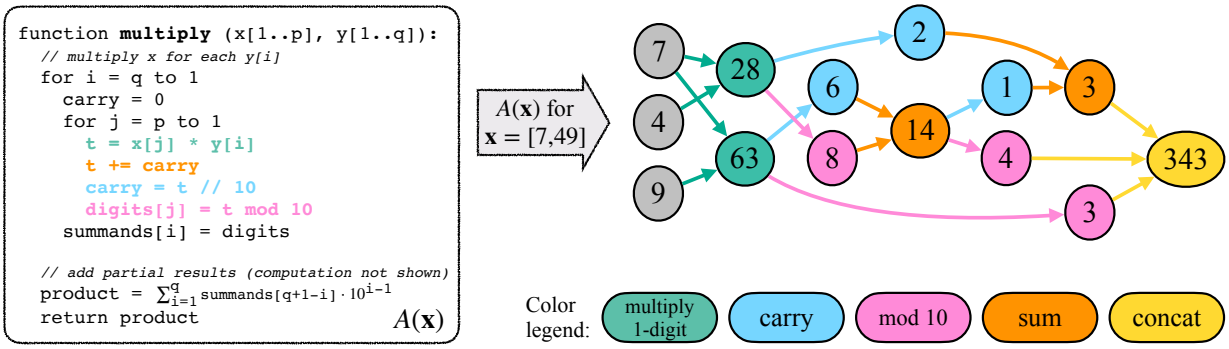


Figure 3.1: Transformation of an algorithm A to its computational graph $G_{A(\mathbf{x})}$. The depicted example is of long-form multiplication algorithm A, for inputs $\mathbf{x} = [7, 49]$ (i.e. computing 7×49).

3.2.2 Quantifying Compositional Complexity using Graph Metrics

A's representation as a computation graph $G_{A(\mathbf{x})}$ enables measuring task complexity from many angles.

We define a node $v \in V$'s *layer number* as the length of the longest path from a source node to v in the directed acyclic graph $G_{A(\mathbf{c})}$. We then define the **reasoning depth** as the largest layer number in the graph. In computation graphs, reasoning depth is a proxy for the maximum level of multi-hop reasoning required to solve the task.

Let $d_S : V \to \mathbb{N}_0$ be the shortest distance to any of G's source nodes $S \subset V$. We define the **reasoning width** of a graph as the mode of $\{d(v) : v \in V\}$. This metric aims to measure the maximum number of variables required to maintain in parallel during the computation.

3.2.3 Predicting Surface Patterns through Relative Information Gain

When evaluating model performance, we may observe partially correct answers even in an overall incorrect response. To understand model strategies in these partial successes, we use Relative Information Gain to predict surface patterns that models are likely to recognize. We represent task T as a distribution $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ and measure the amount of (normalized) information gained about an output element Y_j by observing a subset of input random variables $X \subset \{X_1, \ldots, X_n\}$:

$$\text{RelativeIG}(Y_j, X) = \frac{H(Y_j) - H(Y_j|X)}{H(Y_j)} \in [0, 1] \tag{3.1}$$

RelativeIG may be used to analyze the influence of any node in the computation graph (as defined in §3.2.1) with respect to a set of its ancestors; in particular, output nodes with respect to input nodes.

3.2.4 Exploring Two Representative Compositional Tasks: Definitions

Multiplication Multi-digit multiplication requires executing operations with numerical symbols based on procedural rules (Hiebert, 2013). This task has multiple algorithmic solutions; in constructing computation graphs, we use the well-known $O(k_1 k_2)$ long-form multiplication algorithm for computing $x \cdot y$, where x has $k_1 \leq 5$ digits and y has $k_2 \leq 5$ digits in base 10. See §B.1.1 for data construction details.

Let $\mathcal{F}_A = \{\text{one-digit multiplication, sum, mod 10, carry over, concatenation}\}$ in the instantiation of $G_{A(\mathbf{x})}$. Source nodes S are digits of input numbers, leaf node o is the final output, and intermediate nodes v are partial results generated during execution of the long-form multiplication algorithm (see Figure 3.1).

Dynamic Programming Problem Dynamic programming (DP) recursively breaks down complex problems into simpler sub-problems, so problems solved using this technique are compositional. We analyze a classic relaxation of the NP-complete Maximum Weighted Independent Set problem (Kleinberg and Tardos, 2005): *Given a sequence of integers, find a subsequence with the highest sum, such that no two numbers in the subsequence are adjacent in the original sequence.* This relaxation may be solved in $O(n)$ time using DP. See the solution in §B.1.2. In the experiments, we restrict each integer to the $[-5, 5]$ range.

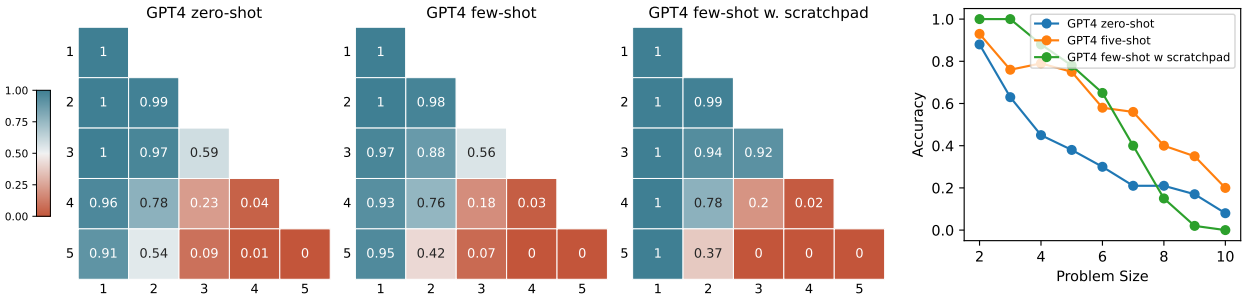
To instantiate $G_{A(\mathbf{x})}$, let $\mathcal{F}_A = \{\text{equals, and, not, indicator function, sum, max}\}$. Source nodes are elements of the input list, and the output node is a list that for each element indicates whether it should be selected. We select an $O(n)$ algorithm since $G_{A(\mathbf{x})}$'s size is proportional to A's complexity.

3.3 Testing the Limits of Transformers: Empirical Evidence

Experimental Setup To understand the capabilities of LLMs, we evaluate GPT3 (`text-davinci-003`) (Brown et al., 2020), and GPT4 (`gpt-4`) (OpenAI, 2023) using zero-shot, few-shot, and finetuning techniques. To understand the capabilities of LLMs, we evaluate GPT4 (`gpt-4`) (OpenAI, 2023) using zero- and few-shot techniques, and GPT3 (`text-davinci-003`) (Brown et al., 2020) using fine-tuning techniques. GPT3 is the most capable OpenAI model that supported fine-tuning at the time the study was performed.

To enable the generation of computation graphs beyond the final answers, we use the concept of *scratchpads* (Nye et al., 2022). Scratchpads are a verbalization of the computation graphs (i.e., a linearized representation of a topological ordering of $G_{A(\mathbf{x})}$). Overall, we consider *question-scratchpad* and *question-answer* (without scratchpad) formats for few-shot and finetuning settings to gauge models' capabilities for learning with and without explicit reasoning. See details of additional models and experimental configurations in §B.2 and examples of scratchpad in §B.1.

3.3.1 Testing the Limits of Transformers with Zero-shot, Few-shot and Finetuning



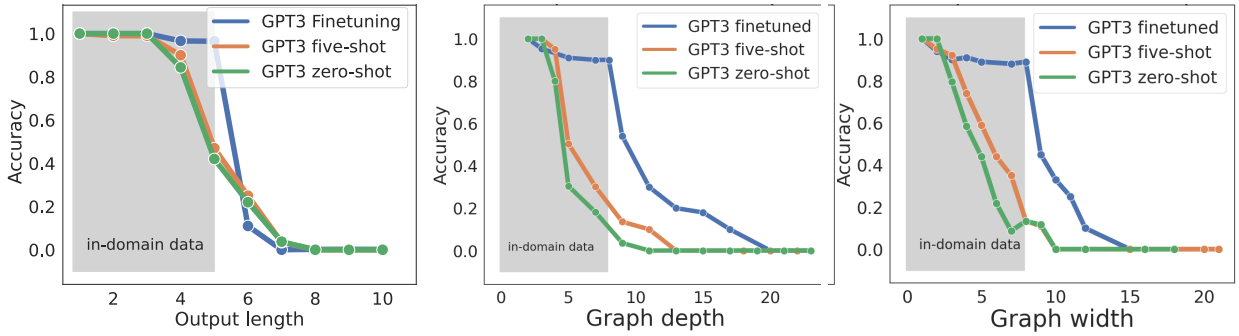
(a) Multiplication (b) Dynamic prog. (DP)

Figure 3.2: **Zero- and few-shot accuracy (without and with scratchpad)** for GPT4 on both tasks. Axes refer to problem sizes (number of digits in each multiplication factor, and sequence length in the DP task). Transformers' accuracy decreases to near zero as task complexity increases, measuring task complexity by the problem size.

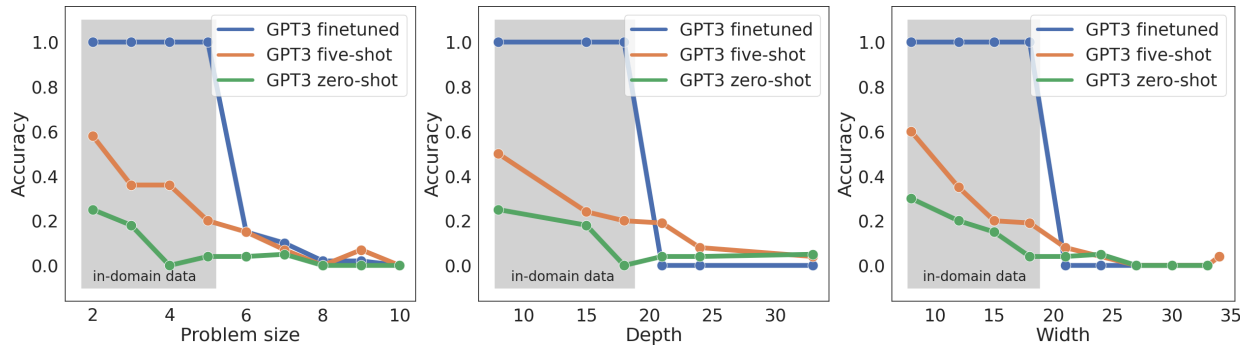
Limits of transformers in zero- and few-shot settings To investigate the inherent problem-solving capabilities of LLMs, we begin by analyzing models' zero-shot and few-shot performances on our compositional tasks. As shown in Figure 3.2, task performances deteriorate significantly from near perfection to zero with increasing complexity, measured by problem size. These results indicate that foundation models' training for GPT4 is not sufficient to teach models how to combine basic operations to solve compositional problems, especially as problems grow more complex.

Limits of transformers with explicit scratchpad training We test whether we can explicitly teach models the required computational operations via *scratchpads*. To do so, we finetune GPT3 with question-scratchpad pairs for both tasks[2]. The results, presented in Figure 3.3, show that GPT3 achieves near-perfect performance on in-distribution, but fails entirely in generalizing to OOD cases—longer outputs, wider or deeper computation graphs. These results indicate that even when training directly with guidance on the computation steps, models still fail to learn component operations in a generalizable manner. Similarly, prompting transformers with question-scratchpad pairs enhances the performance compared to the zero-shot setting, but this performance boost diminishes to zero as instance complexity increases. These findings suggest that the autoregressive characteristic of transformers, which forces them to tackle problems sequentially, presents a fundamental challenge that cannot be resolved by instructing the model to generate a step-by-step solution. Instead, models depend on a greedy process of producing the next word to make predictions without a rigorous global understanding of the task.

[2]We consider all k_1-by-k_2 digit multiplications with $1 \leq k_1, k_2 \leq 4$ and $k_1 \cdot k_2 \leq 9$; and all DP problems up to 5 elements. We selected sizes based on budget constraints for GPT3 finetuning.



(a) **Multiplication.** Training covered problems from 1-digit by 1-digit to 3-digit by 2-digit multiplication, with up to 5 output digits (left panel) and graph depth and width of 8 (center and right panels).



(b) **Dynamic programming (DP).** Training covered problems with sequence length up to 5, and the graph's depth and width were set to 18 (center and right panels).

Figure 3.3: **GPT3 finetuning accuracy on question-scratchpad pairs across different data splits.** The **gray** region represents in-domain data. Although in-distribution performance is near-perfect, GPT3 fails to generalize to out-of-distribution cases, whether measured by output length (left panels), graph depth (center panels), or graph width (right panels).

3.3.2 Breaking Down Successes and Failures of Transformers

3.3.2.1 Information Gain Explains Where Transformers Partially Excel

At times transformers predict partially correct answers even when the overall response is incorrect. We speculate that this may be due to particularities in the task distribution that allow for guessing partial answers without performing the full multi-step reasoning that the task requires.

Using relative information gain (defined in §3.2.3), we can predict surface patterns that a model is likely to learn and contrast them empirically. For multiplication, relative information gain shows that the first digit (two digits) of the output highly correlates with the first digit (two digits) of each input number (see §B.3.1). Hence, this spurious pattern is likely to be learned by a model. Similarly, the prediction of the last digit (or two digits) of the output is observed to solely rely on the last digit (or two digits) of each input number. This pattern holds true due to the principles of modulo arithmetic, which ensures the validity

of this relationship in all cases. Empirically, we verify that models indeed learn the patterns we predicted and other patterns as well (e.g., order of magnitude of the answer, number of trailing zeros for multiplication) in all the settings with and without scratchpad. See details for multiplication, plus dynamic programming task analysis in §B.3.

These experiments suggest that if an output element heavily relies on a single or a small set of input features, transformers are likely to recognize such correlation during training and directly map these input features to predict the output element in testing, without going through the rigorous multi-hop reasoning and giving a false illusion of performing compositional reasoning.

3.3.2.2 Transformers Reduce Multi-Step Compositional Reasoning into Linearized Subgraph Matching

We now explore whether models' correct predictions on unseen test data are due to learning the underlying algorithm or, instead, explainable by exposure to similar training examples. We hypothesize that, beyond simple memorization, transformers largely rely on pattern matching for solving these tasks. To test this, we calculate the average frequency with which partial computations needed to solve an instance appear in the training data, for both correctly and wrongly predicted examples. This analysis requires access to explicit computation graphs, so we use models finetuned with *question-scratchpad* pairs.

Given a model-generated computation graph $\widehat{G}_{A(\mathbf{x})}$ we analyze how often the full computation of each node $v \in \widehat{V}$ is seen in training. We define v's *full computation* as the subgraph induced by all ancestors of v including v, denoted $FC_{\widehat{G}_{A(\mathbf{x})}}(v)$. We say that $FC_{\widehat{G}_{A(\mathbf{x})}}(v)$ is seen during training if $FC_{\widehat{G}_{A(\mathbf{x})}}(v) \cong FC_{G_{A(\mathbf{x}')}}(w)$ for some computation graph $G_{A(\mathbf{x}')}$ in training, and for some $w \in V$. We characterize complexity of a full computation subgraph by its depth, as defined in §3.2.1.

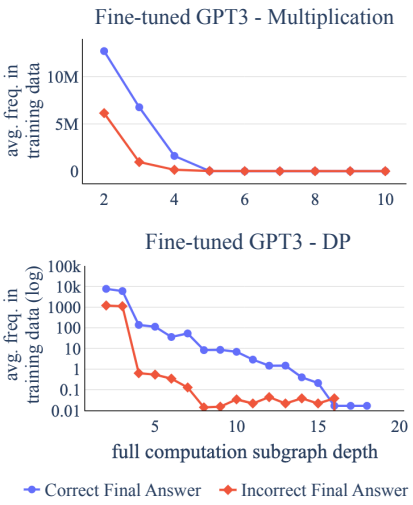


Figure 3.4: Average frequency in which test examples' full computations subgraph appear in the training data w.r.t. the subgraph depth, grouped by final answer.

Figure 3.4 shows that full computation subgraphs appear significantly more frequently in the training data for correctly predicted test examples than for incorrectly predicted ones, for both the multiplication and DP task (both frequencies tend to zero for large depths since we ensured a disjoint train/test split). This high correlation suggests that pattern matching—and not general reasoning capabilities—may be the cause behind correct model outputs. This type of learning could be largely effective when the compositional complexity of tasks is low but it becomes less efficient when tasks are increasingly complex. This may elucidate the observed performance gain in low-complexity and in-domain cases and the striking performance drop in OOD and highly complex cases.

3.4 Error Propagations: The Theoretical Limits

Experiments (§3.3) highlight the limitations of current transformers in handling complex, multi-step reasoning tasks. Concretely, we show that errors rapidly escalate as the problem size grows. Here, we aim to provide theoretical insights into why autoregressive transformer LLMs can perform significantly worse in compositional tasks as the problem size increases, making explicit the different ways in which compounding stochastic errors affect final performance. We argue using stylized examples that transformers may be too limited to solve compositionally complex tasks. Formal statements and full proofs are provided in §B.4.

Algorithms designed to solve compositional tasks typically involve multiple independent applications of a function and/or iterated applications of the same function. A transformer executing such an algorithm acts as an estimator of these functions. In this context, we examine the probability of such an estimator reaching the correct answer as the problem size increases. We first consider a scenario where a transformer estimates an algorithm requiring n independent applications of a function:

Proposition 3.4.1 (informal). *Let f_n involve the combination h_n of n independent applications of a function g. Let $\widehat{f}, \widehat{g}, \widehat{h}_n$ be their estimators. Assume that $\widehat{h}_n$ is a perfect estimator of h_n and that h_n has low collision, with c_n being an upper bound of h_n's collision rate ($c_n < c\ \forall n$, with $c \ll 1$). If $\mathbb{P}(g \neq \widehat{g}) = \epsilon > 0$ where $\widehat{g}$'s errors are independent, then $\mathbb{P}(f_n \neq \widehat{f}_n) > 1 - c_n - (1-\epsilon)^n \cdot (1 - c_n)$. This implies that $\mathbb{P}(f_n \neq \widehat{f}_n)$ decreases **exponentially** as n increases, with $\liminf_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \geq 1 - c$. Moreover, if $c_n \leq \beta\alpha^n$ for some $\alpha \in (0,1), \beta > 0$, $\mathbb{P}(f_n \neq \widehat{f}_n)$ tends exponentially to 1 as n increases.*

Prop. 3.4.1's proof (§B.4.1) shows the rate of convergence is exponential, thus concluding that transformers will rapidly fail with increasing n. Let's now analyze the iterated application function scenario.

Proposition 3.4.2 (informal). *Let $f_n(\mathbf{x}) = g^n(\mathbf{x})$ involve the repeated application of g. Assume that the probability of recovering from a mistake due to the randomness of applying the estimator on an incorrect input has probability at most c. If $\mathbb{P}(g \neq \widehat{g}) = \epsilon > 0$, then $\mathbb{P}(f_n \neq \widehat{f}_n)$ decreases **exponentially** with n. Precisely, $\mathbb{P}(f_n \neq \widehat{f}_n) \geq 1 - (1-\epsilon-c)^{n-1}(1-\epsilon-c/(c+\epsilon))$, implying $\liminf_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \geq 1 - c/(c+\epsilon)$.*

The argument is as follows. Let $s_n := \mathbb{P}(f_n = \widehat{f}_n)$, where $s_1 = 1 - \epsilon$ by definition. Derive $s_n \leq (1 - \epsilon - c) \cdot s_{n-1} + c$ using law of total probability. Then, prove by induction a non-recursive upper bound for s_n with limit $\frac{c}{c+\epsilon}$ when $n \to +\infty$. See formal statement and derivation in §B.4.2.

Prop. 3.4.2's proof also shows an exponential rate of convergence. Note that if $c \ll \epsilon$ then $\liminf_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \approx 1$. It is reasonable to assume $c \ll \epsilon$ when g has low collision, since c represents the probability of the estimator $\widehat{g}(y)$ arriving at the correct output $g(x)$ by chance when given the wrong input $y \neq x$. More details in §B.4.3.

Moreover, repeated applications of a function often imply unbounded errors: if $g(x)$ can be expressed as an affine transformation $Fx+c$, then it may be viewed as a first-order vector autoregression, which are known to be unstable when $|\lambda| \geq 1$ for at least one λ eigenvalue of

F (Hamilton, 1994, Prop. 10.1). While we make these arguments with affine maps, similar behaviors, possibly even more acute, could occur with nonlinear maps (Douc, Moulines, and Stoffer, 2014)—but their study is beyond the scope of this paper.

In Prop. 3.4.2's current form, we implicitly assume that there is a single valid reasoning for each input since g is a function. We can potentially generalize this assumption with a state-transition framing, where the probability of transitioning from a valid state to an invalid one is ϵ, and the probability of recovering from an invalid state is at most c. See formal statement in B.4.2.

All tasks evaluated in the present work can be seen as instances of the results just proven. Prop. 3.4.1 directly applies to multiplication, since m-by-n digit multiplication can be seen as n independent instances of m-by-1 digit multiplication (see Cor. B.4.1). Prop. 3.4.2 directly applies to the recursive function of the dynamic programming task, as well as to m-by-1 digit multiplication, and to the puzzle through its elimination function (details in B.4.3). They are also all low collision settings.

Note that Prop 3.4.1 and 3.4.2 apply to any high-performant estimator of reasoning tasks. We focus on out-of-the-box transformers to align with the scope of our experiments and with the goal of framing empirical results. In §3.5, we discuss how these propositions may inform future research directions.

3.5 Discussion

Collapsed Compositionality and Robustness Implications Transformers today demonstrate undeniably powerful empirical results. Yet, our study suggests that they may have fundamental weaknesses in certain intellectual tasks that require true multi-step compositional operations such as multiplications and dynamic programming. Our careful study based on the computation graph and analyses demonstrates that transformers can often solve multi-step compositional problems by collapsing the depth of the compositional operations via analogical pattern matching. More broadly, our findings suggest that the strong performance of transformers should be taken with a certain grain of salt: Despite initially appearing challenging, certain tasks may not possess the inherent compositionality they seem to have. This is due to the fact that desired solutions could be readily derived from input-output sequences present in the training data, allowing for shortcut pattern matching to produce acceptable solutions. However, such an approach can ultimately result in poor generalization as shown in our study. For example, fine-tuning GPT3 on our tasks both with and without explicit reasoning graphs shows that models' learning fails to generalize beyond levels of complexity seen in training.

Theoretical Findings and their Empirical Implications The proofs presented in §3.4 show that, under reasonable assumptions, the probability of incorrect predictions converges exponentially to ≈ 1 for abstract compositional tasks. Importantly, these proofs apply to autoregressive LMs in general. Our insights indicate that the current configuration of transformers, with their reliance on a greedy process for predicting the next word, constrains their error recovery capability and impedes the development of a comprehensive global understanding of the task. Building on these findings, we suggest several empirical strategies

for harnessing the potential of transformers. Firstly, transformers may be employed in ways that require chaining only a few compositional steps to reach a solution rather than lengthy reasoning steps (e.g., (A. Q. Jiang et al., 2023)). Secondly, transformers may be best suited for compositional tasks where evaluation metrics can afford some leniency; for example, finding approximate solutions that do not require executing the whole graph, such as identifying the most significant digit in a multiplication. Finally, we suggest augmenting transformers with planning modules as well as using refinement methods, that can iteratively improve their generations (Madaan et al., 2023; Welleck et al., 2023).

3.6 Related Work

Reasoning abilities in transformer LLMs Recently, transformers (Brown et al., 2020; Chowdhery et al., 2023; Chung et al., 2024; OpenAI, 2022, 2023; Rae et al., 2021; Taylor et al., 2022; Thoppilan et al., 2022) have demonstrated impressive reasoning abilities across a wide range of tasks, even outperforming humans in certain cases (Choi et al., 2023; H. Fu Y. P. and Khot, 2022; J. Wei et al., 2022b; Zheng and Zhan, 2023). This success has been largely attributed to the scaling effect, where larger models and training datasets result in improved performance (Aghajanyan et al., 2023; Henighan et al., 2020; Kaplan et al., 2020). However, these models have also been shown to struggle across multiple domains (Helwe, Clavel, and Suchanek, 2021), including algorithmic reasoning (Veličković and Blundell, 2021), commonsense reasoning (Koralus and Wang-Mascianica, 2023; C. Qin et al., 2023), theory of mind (Sap et al., 2022), planning (Valmeekam et al., 2022), logical reasoning (Srivastava et al., 2023), and ethical reasoning (L. Jiang et al., 2021). These difficulties have motivated us to take a step back and thoroughly examine both the successes and failures of transformers from empirical and theoretical perspectives on compositional reasoning tasks.

Challenges of transformers in compositional tasks Transformers perform fairly well in single-step reasoning tasks (Srivastava et al., 2023), but face challenges when it comes to effectively combining multiple steps to solve compositionally complex problems (Nye et al., 2022; Saparov and He, 2023; Welleck et al., 2022a; Y. Zhang et al., 2022). Recent research has focused on overcoming these limitations through various approaches. First, fine-tuning transformers to directly generate the final answer while keeping the reasoning implicit (Betz, Voigt, and Richardson, 2021; P. Clark, Tafjord, and Richardson, 2021). Second, encouraging transformers to generate reasoning steps explicitly within a single generation (Liang et al., 2023; Ling et al., 2017; Nye et al., 2022; J. Wei et al., 2022a). For example, Nye et al. (Nye et al., 2022) and Zhou et al. (H. Zhou et al., 2023) used scratchpads to teach transformers how to perform algorithmic reasoning tasks such as addition by splitting the task into intermediate steps (Ling et al., 2017; J. Wei et al., 2022a). Further, leveraging LLMs to generate each reasoning step iteratively via a selection and inference mechanism (Creswell and Shanahan, 2022; Creswell, Shanahan, and Higgins, 2023; Tafjord, Dalvi, and P. Clark, 2021). Lastly, choosing a training split that maximizes the number of observed patterns between the train and test data (Bogin, S. Gupta, and Berant, 2022), or diversifying in-prompt examples to cover the maximum of patterns (Levy, Bogin, and Berant, 2023), ultimately enhancing generalization. The primary focus of these studies is to enhance model performance on

compositional problems without striving for complete mastery. In contrast, our work explores the fundamental limits of vanilla transformers in achieving full mastery, striving for 100% performance in both in-domain and OOD settings. Our findings show that reaching full mastery is inherently challenging, providing insights into the complexities involved.

Challenges of transformers in generalization Extensive research has been done to investigate the generalization capabilities of transformers (Anil et al., 2022; Bhattamishra, Ahuja, and Goyal, 2020; Dubois et al., 2020; B. Liu et al., 2023; Newman et al., 2020; Razeghi et al., 2022). This encompasses various facets of generalization, including easy-to-hard generalization (Bansal et al., 2022; Schwarzschild et al., 2021), length generalization (Anil et al., 2022; Bhattamishra, Ahuja, and Goyal, 2020; Bueno et al., 2022; Newman et al., 2020; Press, Smith, and Lewis, 2022), and generalization on symbolic mathematical integration (Welleck et al., 2022b). Schwarzschild et al. (Schwarzschild et al., 2021) and Bansal et al. (Bansal et al., 2022) employ weight-tied neural networks to generalize from easy to hard examples. Liu et al., (B. Liu et al., 2023) found that shallow transformers learn shortcuts during training, leading to poor OOD generalization. Razeghi et al. (Razeghi et al., 2022) revealed a positive correlation between the frequency of training terms and their test performance. Building upon this line of inquiry, we present a more rigorous examination of sub-graph matching between training and test instances for complex compositional tasks where we demonstrate how pattern matching can hinder generalization. We complement our empirical results with theoretical insights on transformers' limits.

Transformers' theoretical expressiveness Lin et al. (C.-C. Lin et al., 2021) study autoregressive models' limitations from a computational complexity theory perspective. Transformer-specific work has focused on quantifying the class of problems that (not necessarily autoregressive) transformers can express assuming perfect parameters (Chiang, Cholak, and Pillay, 2023; Merrill and Sabharwal, 2023a,b, 2024, inter alia). All tasks analyzed in our work belong to a class expressible by transformers, suggesting that known upper bound might not be tight. Importantly, Hahn (Hahn, 2020) shows that transformers cannot robustly model noncounter-free regular languages even when allowing infinite precision. In contrast, our focus is on error accumulation, which enables to investigate if reasoning tasks theoretically solvable by transformers are likely to be solved by them.

Iterated Functions The process of repeatedly applying a noisy single operation or function f can be related to iterated random functions (Diaconis and Freedman, 1999). In this latter literature, the focus is usually on the contractive regime in which accrued errors can be kept under control, and the subsequent convergence guarantees (e.g., Delyon and Juditsky, 1999). When f is an affine transformation, the process falls simultaneously between two perspectives: time series (Hamilton, 1994) and dynamic programming and control (Bertsekas, 2022). We leverage the former to discuss the often explosive errors of f^n.

3.7 Conclusions

On a broader scope, as transformers continue to gain widespread deployment with significant real-world impacts, it is ever more urgent to understand their successes and failures. Our study critically investigates transformers' limitations and emphasizes the need to develop models capable of robust generalization and systematic problem-solving. By examining the compositional capabilities of these models, we aspire to work towards more reliable AI systems that excel not only in tasks where abundant training examples are sufficient, but also in cases requiring precise compositional reasoning.

3.8 Compositionality: A 2026 Perspective

The study just presented involved frontier models at the time of the study in May 2023. Since then, LLM capabilities have improved substantially, motivating a revisit of model performance in these tasks as well as evaluating larger problem sizes, and a new class of *reasoning* models that produce extended internal chain-of-thought before answering. We re-evaluate on the DP and multiplication tasks using some of the latest Gemini models as of April 2026: Gemini 3.1 Flash Lite (thinking disabled; asking the model to follow the scratchpad shown) as a non-reasoning baseline, and Gemini 2.5 Flash, Gemini 3 Flash Preview, and Gemini 3.1 Pro Preview with thinking enabled.

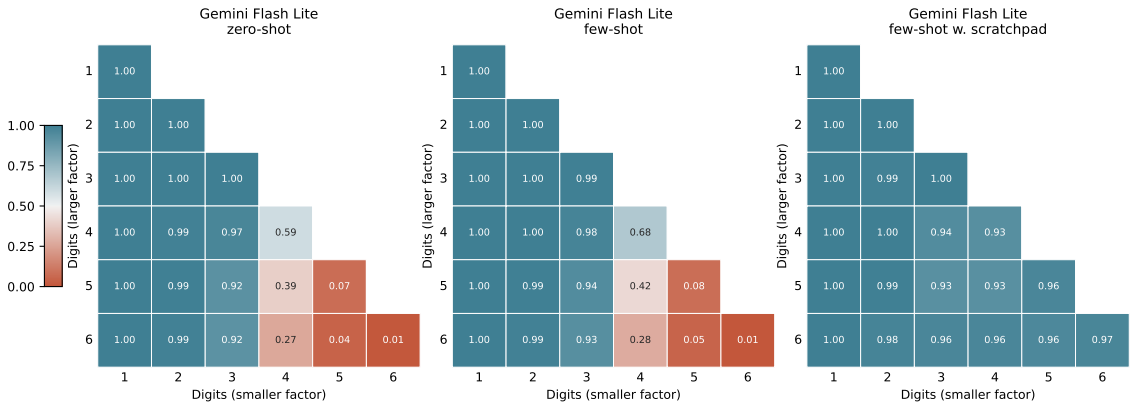


Figure 3.5: Multiplication accuracy for Gemini 3.1 Flash Lite across digit counts (lower-triangular heatmaps). Scratchpad prompting now sustains near-perfect accuracy through 6 digits—significantly further than GPT-4 in 2023.

Scratchpad-following capabilities have improved, length generalization may not: a study on multiplication As shown in Figure 3.5, models have become much better at *following* a provided scratchpad algorithm, as well as robustly memorizing a larger pool of smaller instances. This robust memorization also has positive consequences at larger problem sizes, since the error propagation described in Section 3.4 will delay its occurrence.

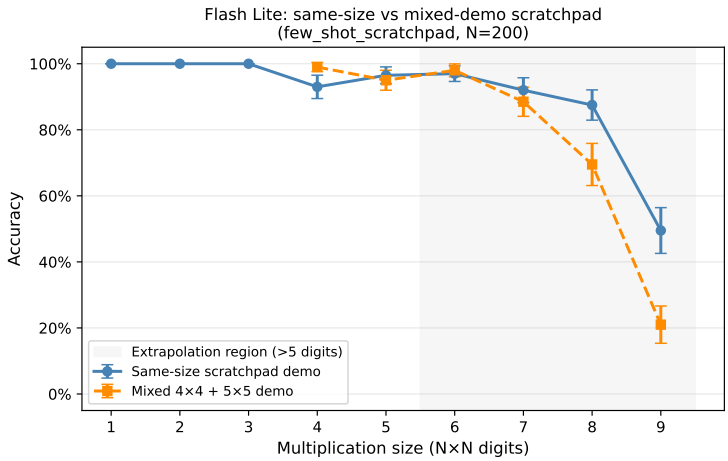


Figure 3.6: Multiplication accuracy for $N \times N$ multiplication when showing same-size scratchpad demonstrations, vs. when showing mixed 4-digit by 4-digit plus 5-digit times 5-digit multiplication for Gemini 3.1 Flash Lite. Accuracy decays more sharply when the demonstrations are not of the same problem size as the evaluation, even when different problem sizes are shown to encourage length generalization.

Figure 3.6 also shows that scratchpad-following capabilities are still present at larger problem sizes than ever before, sustaining near-perfect accuracy through 6 digits and degrading gracefully to 50% at 9 digits. However, Figure 3.6 also points out that the generalization capability may still be limited. Concretely, when the few-shot examples show a mix of 4-digit by 4-digit multiplications, and 5-digit by 5-digit multiplications, instead of the exact problem size required at testing, model performance decays significantly, dropping to 20% at 9 digits vs. 50% when using the exact same problem size for few-shot examples. This suggests that the clear improvements in procedural fidelity over what we observed with GPT-4 are still lacking the length generalization capabilities we would expect from a model genuinely understanding the underlying algorithm.

We also directly compare scratchpad (thinking disabled) against scratchpad combined with thinking, using Gemini 2.5 Flash. At 4 digits, scratchpad alone achieves 93% while scratchpad + thinking drops to 79%; at 5 digits the gap widens to 97% vs. 12%. This suggests that showing extremely relevant few-shot examples is still more beneficial than enabling free-form thinking when these examples are available, even for such a well-known task as multiplication.

Free-form thinking capabilities are effective, yet problem size degradation persists as in the original work Finally, we also study the effects of free-form thinking in the other task of interest, DP. Figure 3.7 shows that Gemini models handle much larger DP instances than the original GPT-4 results, with the most capable model, Gemini 3.1 Pro Preview, reaching 100% at $n = 10$. However, the degradation pattern described in this chapter still reappears, just at larger sizes—40% at $n = 15$, 29% at $n = 20$, 25% at $n = 25$—and smaller models degrade even faster. Extended reasoning delays but does not eliminate the compositional ceiling.

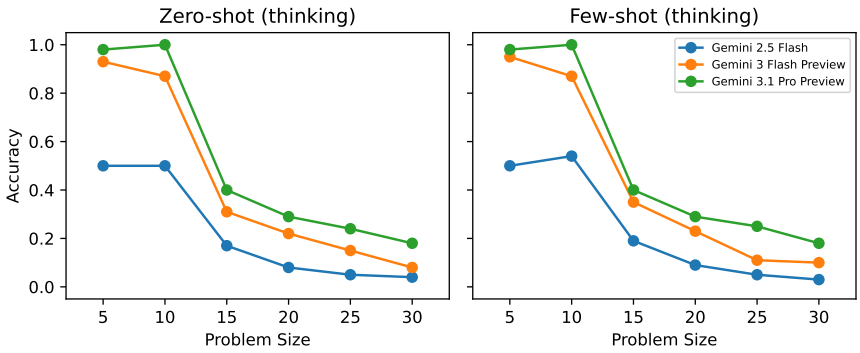


Figure 3.7: DP accuracy vs. problem size for three thinking-enabled Gemini models (zero-shot left, few-shot right; thinking enabled, no scratchpad). Performance degrades sharply beyond $n = 10$ for all models despite our problem being a classic DP problem.

In summary, both experiments point to similar compositional reasoning decay behavior as in 2023, but pushing the base error rate significantly further to enable much larger problem sizes—which may be sufficient for many real-world applications. As previously mentioned, the tasks chosen for our study are intentionally simple and should never be understood as sufficient test for generalizable AI capabilities.

Part III

Modeling Semantic Structure: Theory of Mind Reasoning

3.9 Introduction

Part II demonstrated that imbuing tasks with explicit semantic-level structure is a productive strategy for principled analysis of language model behavior: by recasting reasoning problems as computation graphs, complexity becomes formally quantifiable and intermediate steps independently verifiable, revealing principled limits on compositional generalization in transformers. Importantly, all tasks considered were mathematical in nature. Part III asks whether semantic-level structure strategies can be extended to *Theory of Mind* (ToM) reasoning (Premack and Woodruff, 1978)—the capacity to reason about other agents' beliefs, desires, and intentions—a domain that is social rather than mathematical, and where models must track the shifting mental states of multiple agents across a narrative. This ability is a cornerstone of human social intelligence, developing naturally in children (C. Frith et al., 2003) and underlying reading comprehension, dialogue, negotiation, and virtually every form of collaborative interaction. Equipping language models with robust theory of mind capabilities is therefore a prerequisite for effective deployment across a wide range of applications.

Theory of mind presents two interconnected empirical challenges for language model research: reliably measuring the extent and limits of models' theory of mind capabilities, and improving models so they more reliably exhibit them. Both are addressed by the chapters that follow.

On the evaluation side, the scientific literature has reached conflicting conclusions about the extent to which LLMs exhibit theory of mind behavior. Prior work has alternately interpreted model outputs as evidence of emerging ToM-like capabilities (Bubeck et al., 2023; Kosinski, 2024) or as evidence of their absence (Sap et al., 2022; Shapira et al., 2024; Strachan et al., 2024; Ullman, 2023), and these conflicting interpretations are driven in large part by the difficulty of designing evaluations that isolate mental state modeling from other cues. The dominant evaluation paradigm—pioneered by ToMi (Le, Boureau, and Nickel, 2019) and ToM-bAbI (Nematzadeh et al., 2018)—relies on template-based datasets that cover a restricted set of interaction types and scenario structures, that are sometimes unusual or violate commonsense physical reasoning . Datasets in this paradigm are often susceptible to being solved through spurious pattern-matching rather than systematic mental state modeling (Ullman, 2023). Human-generated alternatives, while richer, face their own limitations: annotation cost constrains scale, and the diversity of covered scenarios is itself bounded by the creativity and effort that annotation pipelines can sustain. The rapidly increasing capability of frontier models means that benchmarks risk saturation at the time of release. Together, these challenges make it difficult to reliably measure the extent and limits of models' theory of mind capabilities, and harder still to determine how to remedy failures once they are identified.

On the improvement side, prior work has primarily relied on supervised training (e.g., Arodi and Cheung, 2021; Grant, Nematzadeh, and Griffiths, 2017; Nematzadeh et al., 2018), but the simplicity and limited diversity of available datasets meant that such models were brittle, failing under even slight out-of-distribution perturbations—as will be demonstrated in Chapter 4. While Part II established that transformers face fundamental limits on complex reasoning tasks, the empirical question of how far fine-tuning methods can push theory

of mind capabilities in practice remains valuable to answer. Existing data sources are furthermore ill-suited for use as training data: they are often limited in scale (Xu et al., 2024), portray specific scripted scenarios (Le, Boureau, and Nickel, 2019; Wu et al., 2023), and are prone to leakage risks that would make them fully unsuitable for future use. The quality and diversity of available data thus emerges as the core tractable bottleneck.

We explore two responses to this bottleneck. The first, pursued in Chapter 4, is to sidestep the data problem entirely: by imposing explicit belief-graph structure at inference time, models can reason more reliably about mental states without further training, yielding a solution more general than any supervised approach trained on the limited data available at the time. The second, pursued in Chapter 5, is to address the data bottleneck directly: taking inspiration from the structured belief representations of Chapter 4, it introduces an A*-powered algorithm that generates diverse, challenging, and verifiable theory of mind stories adversarially tailored to expose the weaknesses of a target model, making supervision viable in a way it was not before. Crucially, this structured generation approach simultaneously addresses both challenges: the same adversarially generated data that trains stronger models also yields evaluation sets that are more diverse and more robust to saturation, offering a principled path through the conflicting claims that have characterized the evaluation literature.

Together, the two chapters will show that structure-guided approaches can push model performance on social reasoning substantially further than the status quo, while also providing the tools needed to keep probing where the remaining limits lie.

Chapter 4

A Plug-and-Play Multi-Character Belief Tracker for Theory of Mind Reasoning

*This chapter is based on **SymbolicToM**: Minding Language Models' (Lack of) Theory of Mind: A Plug-and-Play Multi-Character Belief Tracker. Published at ACL 2023 (Outstanding Paper Award).*

4.1 Introduction

Cognitive and literary studies have extensively argued theory of mind's key role in understanding stories, in order to explain and predict each character's actions (Carney, Wlodarski, and Dunbar, 2014; Duijn, Sluiter, and Verhagen, 2015; Leverage, Mancing, and Schweickert, 2010; Zunshine, 2006, inter alia). As exemplified in Figure 4.1, readers need to model Bob's mental state (called *first-order* theory of mind), as well as Bob's estimation of Alice's mental state (*second-order* theory of mind) to answer questions.

As discussed in §3.9, the scarcity and limited diversity of available theory of mind data makes supervised approaches brittle—motivating an inference-time solution instead. We introduce SymbolicToM, an inference-time method that improves large language models' theory of mind capabilities by augmenting them with an explicit symbolic graphical representation of each character's beliefs. Unlike prior efforts, our approach does not require training and instead divides the problem into simpler subtasks, leveraging off-the-shelf models to solve them, and carefully consolidating their results. This makes SymbolicToM significantly more robust than existing models trained specifically for theory of mind behavior.

While beliefs about the world state differ among people, most existing work on encoding belief states does not model this behavior, relying on singular graphs (Jacqmin, Barahona, and Favre, 2022; Jansen, 2022). SymbolicToM, instead, utilizes a *set of graphs*, each representing what the character p_1 *thinks that* p_2 *believes that [...]* p_m *assumes to be the current state of the world*, where m is the maximum reasoning depth as determined by the user. This explicit, recursive mental state representation enables the model to answer

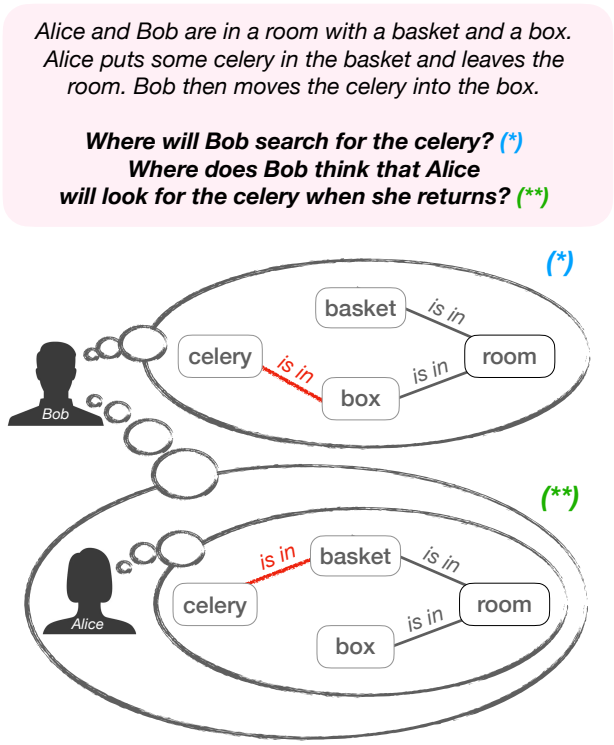


Figure 4.1: A simple story requiring theory of mind. Note that Alice's belief of the celery's location differs from reality (i.e. Alice holds a *false belief*). Readers must reason that Alice will look for the celery where she left it, and that Bob will make that same assumption. Questions shown require different depths of mental state modeling.

questions from the perspective of each character. SymbolicToM's process of selecting and querying a particular character's graph grounds it in cognitive science research arguing theory of mind as an essential mechanism of selective attention (Leslie, Friedman, and German, 2004). Our approach also instills desirable inductive biases, such as object permanence—for example, object locations (represented by edges in the graphs) are assumed to be constant until the method can infer a change. Although existing NLP datasets only test up to second-order reasoning (i.e., $m \leq 2$), SymbolicToM is designed to work at any depth.

SymbolicToM dramatically improves the performance of large language models in theory of mind reading comprehension tasks. For example, GPT-3-Davinci's (Brown et al., 2020)[1] accuracy on the ToMi benchmark (Le, Boureau, and Nickel, 2019) increases by 38 absolute points using SymbolicToM (yielding 92% accuracy averaging across question types). Furthermore, we extend the ToMi test sets with diverse story structures and sentence paraphrases and demonstrate that our approach is significantly more robust than supervised approaches.

[1]The strongest model available at the time of the study, published in July 2023.

4.2 Motivation and Background

Although large-scale language models have recently shown improvements in some classic theory of mind examples, they are still far from reliably showing theory of mind capabilities (Sap et al., 2022; Shapira et al., 2024; Ullman, 2023; P. Yu et al., 2023). While the training data for these models includes human-written stories which require theory of mind reasoning, this information is largely implicit and hence difficult for models to learn. ChatGPT and GPT3-Davinci's incorrect answers to Figure 4.1's question #2 are shown below.[2]

> **ChatGPT `(gpt-3.5-turbo)`:** Based on the information provided, Bob would likely think that Alice will look for the celery in the box when she returns. Since Bob moved the celery from the basket to the box, he would assume that Alice would expect to find it in its new location.
> **GPT3 `(text-davinci-003)`:** Bob will likely think that Alice will look for the celery in the box, since that is where he moved it.

To the best of our knowledge, at the time of this study, the only available datasets for measuring theory of mind in reading comprehension tasks were ToM-bAbI (Nematzadeh et al., 2018) and ToMi (Le, Boureau, and Nickel, 2019).[3] In this work, we focus on *cognitive* theory of mind—differences in knowledge and beliefs among characters—as opposed to *affective* aspects such as intent inference or character consistency (Qiu et al., 2022; P. Zhou et al., 2022), which are the primary focus of text-based game settings. Because of their templated nature, supervised training on these datasets is prone to overfitting to spurious artifacts. While ToMi was developed to counter this behavior in ToM-bAbI by introducing flexible sentence ordering and distractor sentences and characters, we show it still faces the same pitfalls.

As discussed in §3.9, the scarcity and limited diversity of available theory of mind data makes supervised approaches brittle—motivating an inference-time solution instead.

4.3 Methods

4.3.1 SymbolicToM: Algorithm Overview

Our goal is to automatically answer reading comprehension questions given a story involving multiple characters, without requiring any supervised training or fine-tuning on this task. We first introduce key notation, then provide a high-level overview of SymbolicToM (Algorithm 2).

Notation We use the term *k-th order theory of mind* to refer to an estimate of what a character p_1 *thinks that* p_2 *thinks that [...]* p_k *thinks about the world state.* We denote this

[2] Queried on May 22, 2023 with `top_p=1` and `temperature=0`. Given the non-deterministic and continuously changing nature of these models, exact examples may not produce the same response we report.

[3] This reflects the state of available benchmarks at publication (July 2023); subsequent work has introduced additional datasets (e.g., Wu et al., 2023; Xu et al., 2024).

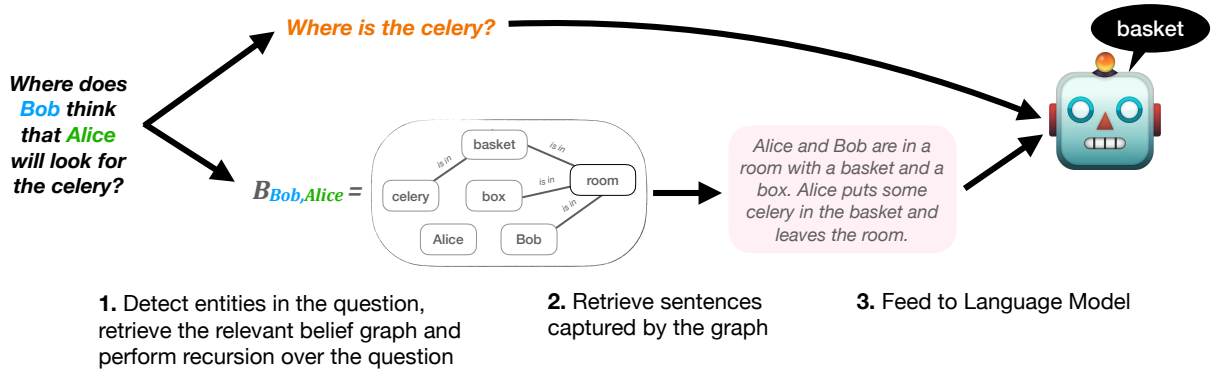


Figure 4.2: Pipeline overview of SymbolicToM, a decoding-time algorithm that enhances large language models' theory of mind capabilities. SymbolicToM does not require training: it divides the problem into smaller subtasks and uses off-the-shelf models to solve them. Given a passage, SymbolicToM constructs explicit symbolic graphical representations of each character's belief states (step 1). To answer ToM questions, it retrieves relevant sentences from the graph (step 2) and then queries the LLM in a zero-shot manner (step 3).

belief by $B_{p_1,\ldots,p_k}$. We let $k \leq m$, where m is a maximum reasoning depth. This is a user-specified limit, denoting the maximum recursion that the reader is assumed to be capable of performing. For instance, in Figure 4.1, questions #1 and #2 measure 1st- and 2nd-order theory of mind respectively; B_{Bob} refers to Bob's beliefs about the current world state, and $B_{\text{Bob,Alice}}$ represents Bob's estimation of Alice's beliefs about the world state. In this work, $B_{p_1,\ldots,p_k}$ only represents beliefs about the current world state, without additional modeling of other characters' mental states, such as their opinions.

A benefit of this notation is that any belief state can be represented as an m-th order one. We assume that *what* p_k *thinks that* p_k *thinks* is equivalent to *what* p_k *thinks*, and by induction, $B_{p_1\ldots p_k} \equiv B_{p_1,\ldots,p_k,p_k,\ldots,p_k}$, where the last p_k is repeated $m-k$ times. We adopt this notation going forward, denoting all states as m-th order. As a conceptual note, the set of belief states $\{B_{p_1\ldots p_k,q_{k+1}\ldots q_m} \mid \forall q_{k+1},\ldots,q_m\}$ represents the mental state from the perspective of $p_1,\ldots,p_k$, using $m-k$ order of theory of mind.

Local and Global Context We represent each $B_{p_1\ldots p_k}$ as a graph (a simplified version is depicted in Figure 4.1) where each node represents an entity (e.g. a character, object, room, container) and each edge connects two nodes with a stated relationship in the story. We construct the graphs by iterating through a story one sentence at a time, and adding both nodes and edges to the graph (BeliefTrackingStructure; described in §4.3.2 and Algorithm 3). Each edge is also paired with the sentence from the story from which it was constructed. We refer to the set of all belief state graphs as the *local contexts*. We also maintain a *global context* graph, denoted by G, which contains the true world state. G has an identical structure to $B_{p_1\ldots p_k}$. See C.1.1 for a detailed definition of G.

Question Answering After parsing a story and constructing the complete set of belief-tracking structures, we can use these structures to answer questions by querying the ap-

propriate graph and considering it as the real-world state. For example, if the question is "Where will Bob think that Alice will look for the celery?", we retrieve $B_{\text{Bob, Alice}}$, but if instead the question were "Where will Bob look for the celery?", we would retrieve B_{Bob}. In both cases, we would ask "Where is the celery?" on the retrieved graph. Figure 4.2 shows an example of the full pipeline.

Given a question, we identify the relevant characters $p_1, \ldots, p_k$ mentioned in order heuristically, and rephrase the question to ask directly about the world state (PROCESSQUESTION; owing to the questions' templatic nature in our evaluation data, this approach rephrases all questions correctly).[4] We then retrieve the corresponding graph; i.e., $B_{p_1,\ldots,p_k}$, of which we can simply ask the question "Where is the celery?". To obtain the answer, we first reconstruct a subset S' of sentences in the original story, consisting of those represented by the retrieved graph (SENTENCESREPRESENTEDBYGRAPH). We then use a large language model $\mathcal{L}$ to answer the simplified question zero-shot given S', using as input the sentences in S' in the same order as they appeared in the original text, and preserving phrasing. We optionally further filter S' based on the entities mentioned in the question (FILTERBASEDONQUESTION). An ablation study showed this last step can often be skipped (see Appendix C.3.1).

Algorithm 2 SYMBOLICTOM

$B \leftarrow$ BELIEFTRACKINGSTRUCTURE(*sentences*)
$p_1, \ldots, p_k, question' \leftarrow$ PROCESSQUESTION(*question*)
$S' \leftarrow$ SENTENCESREPRESENTEDBYGRAPH($B_{p_1,\ldots,p_k}$)
$S'' \leftarrow$ FILTERBASEDONQUESTION($S', question$)
return $S'', question'$

4.3.2 Computing the Belief Graphs $B_{p_1 \ldots p_k}$

Assuming each story is told chronologically, SYMBOLICTOM processes each sentence s sequentially in two stages (Algorithm 3). First, it extracts all actions in s and updates the global context G from an omniscient point of view while identifying the characters ($\mathcal{W}$) who witnessed actions and world state changes described in the sentence. Second, for each witness $w \in \mathcal{W}$, it propagates this new information to update w's local contexts; i.e., we only update $B_{p_1,\ldots,p_m}$ with, for $1 \leq i \leq m$, each $p_i \in \mathcal{W}$, and leave the rest unchanged.

As an example, when processing the last sentence in Figure 4.3, we update Bob and Charles's state (B_{Bob} and B_{Charles}) and the perception of others' respective state ($B_{\text{Bob,Charles}}$, $B_{\text{Charles, Bob}}$), but we need not update Alice's state, or Bob and Charles's perception of Alice's mental state, because she did not witness the actions described.

4.3.2.1 Detecting Witnesses, Updating Graphs, and Propagating Knowledge

Starting with an empty graph, for each new sentence s, we update the global context G by combining off-the-shelf models in four steps (Algorithm 4; GLOBALCONTEXTUPDATE).

[4] Our explorations show that GPT3 is also capable of rephrasing the questions zero-shot (see §C.1.3), but we refrained from this solution due to budget concerns.

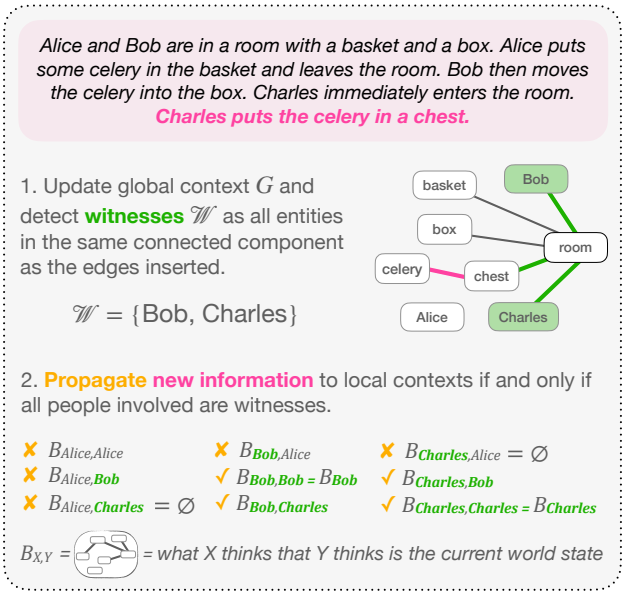


Figure 4.3: High-level depiction of the belief update procedure for $m = 2$. $B_{p_1,\ldots,p_k}$ denotes a graph, and the graph updating procedure is detailed in the main text.

Algorithm 3 Belief Tracking

```
function BELIEFTRACKINGSTRUCTURE(sentences)
    for s ∈ sentences do
        G, 𝒲 ← GLOBALCONTEXTUPDATE(G, s)
        for all [p_1, ..., p_m] ∈ 𝒲^m do
            B_{p_1...p_m} ← LOCALCONTEXTUPDATE(B_{p_1...p_m}, G, s)
```

First, we detect the existing edges E in G that contradict s. This is implemented as detecting Natural Language Inference (NLI) contradictions, considering s as the premise, and every edge in G as a hypothesis.

Second, we augment G with new edges and nodes, by first deriving a natural language representation r of the state resulting from the actions described in s, and then extract new nodes and edges from r as OpenIE triples (Stanovsky et al., 2018). For example, for "Bob then moves the celery to the box", the resulting state r would be the sentence "The celery is in the box". To obtain r from s, we prompt a language model such as GPT3 (see Appendix C.1.2 for details). After obtaining r, we use the corresponding triple (e.g., `(celery, box, is in)`) to add new nodes and edges to G if not already present (e.g., the nodes "celery" and "box", and a directed edge connecting them labeled by "is in"). Importantly, we only add edges that represent positive relations between nodes; i.e., there will not be an edge representing "The celery is not in the box".

Third, we detect the witnesses $\mathcal{W}$ of the actions described in s. Since each character will be a node in G, we identify $\mathcal{W}$ as all the characters that are in the same connected component as the newly added edges.

Finally, we remove all edges E that are no longer valid in G as identified by the NLI contradictions. This step is done last to ensure all witnesses are found before their edges are deleted.

Algorithm 4 World State Beliefs Graphs Update

function GLOBALCONTEXTUPDATE(G, s)
  $E \leftarrow$ DETECTCONTRADICTINGEDGES(G, s)
  $G \leftarrow G \cup$ TRIPLES(RESULTINGSTATE(s))
  $\mathcal{W} \leftarrow$ FINDWITNESSES(G)
  $G \leftarrow G \setminus E$
  return $G, \mathcal{W}$

function LOCALCONTEXTUPDATE(C, G, s)
  $E \leftarrow$ DETECTCONTRADICTINGEDGES(C, s)
  $C \leftarrow C \cup$ TRIPLES(RESULTINGSTATE(s))
  $C \leftarrow$ PROPAGATEKNOWLEDGE(G, C, s)
  $C \leftarrow C \setminus E$
  return C

The local contexts ($B_{p_1,\ldots,p_k}$) are updated similarly (LOCALCONTEXTUPDATE in Algorithm 4), except for an additional step of knowledge propagation. While performing an action, a character may implicitly gain information not described in the text. For example, when entering a room, a character may gain knowledge of the people and visible objects in the room. This knowledge (already present in G, which tracks the omniscient world state) needs to be propagated to each $B_{p_1,\ldots,p_k}$ with each $p_i \in \mathcal{W}$. As G represents the true world state, we simplify the problem: if a character p_i is in a specific connected component D of G, then it possesses all knowledge encoded in D. To model implicit knowledge gain, we add all edges in D to $B_{p_1,\ldots,p_k}$. As D represents the latest global context information, we remove from the local context edges that are in $B_{p_1,\ldots,p_k}$ but not in D (representing outdated beliefs about the world state).

4.3.3 Notes on Memory Efficiency

Memory requirements grow exponentially with m, the maximum order of theory of mind considered. However, m in practice is small, as humans find tasks increasingly challenging as m increases. For example, psychological tests for $m = 3$ are aimed at teenagers and adults (Valle et al., 2015). All experiments in this work are done with $m = 2$, the maximum order of theory of mind reasoning that current datasets evaluate. If memory were a concern, one could process the questions first for memory efficiency, and compute only the graphs $B_{p_1,\ldots,p_k}$ required for target queries.

4.4 Fundamental Issues in Existing ToM Datasets

Construction of ToMi As introduced in §4.2, the sole large-scale theory of mind dataset for reading comprehension tasks is ToMi (Le, Boureau, and Nickel, 2019). Barring its added

1. Oliver entered the front yard.
2. Ethan entered the front yard.
3. Liam entered the kitchen.
4. **objectA** is in the basket.
5. Ethan exited the front yard.
6. Ethan entered the kitchen.
7. Oliver moved **objectA** to the **containerX**.
8. Where does Ethan think **objectA** is?
ToMi Gold Label: basket

Table 4.1: Interpretation of ambiguities in ToMi can be affected by commonsense. In the above template, the correct label is that Ethan thinks **objectA** is in the *basket*, as this is where he last saw it. Setting **objectA** to *hat* and **containerX** to *box* results in 80% human accuracy. However, setting these to *apple* and *pantry*, accuracy drops to 20%. Physical commonsense suggests the pantry is likely in the kitchen, changing the answer to *pantry*, but regardless of the identity of **objectA** or **containerX**, the correct label in ToMi is *basket*.

distractor characters and sentences, ToMi strictly mimics the Sally-Anne test, a widely adopted evaluation for assessing children's social cognitive ability to reason about others' mental states (Baron-Cohen, Leslie, and U. Frith, 1985; Wimmer and Perner, 1983). Stories are structured are as follows: characters A and B are in a room, and A moves an object from an opaque container to another; B may or may not leave the room before A moves the object. B will know the object's new location if and only if they were in the room at the time it was moved. Four types of ToM questions are posed: first-order or second-order, probing a character about either a true or a false belief (i.e, belief that matches reality or not). ToMi also includes questions probing about reality (or *zeroth-order* ToM, Sclar, Neubig, and Bisk, 2022) and memory.

ToMi has six types of sentences (i.e. six *primitives*) with set phrasing. These include someone (a) entering or (b) exiting a room; the location of (c) an object or (d) a person; (e) someone moving an object; and (f) someone's opinion about an object (distractors). Primitives are combined into stories with a finite list of possible orderings. Despite the limited types of primitives, correctly answering questions requires high-order levels of reasoning.

Templated stories are filled with randomly sampled objects, locations, containers, and rooms from a set list. ToMi implicitly assumes that questions about the story do not depend on these decisions, only on the underlying story template. Yet, in a small-scale human study, we find physical commonsense leads human answers to change, and disagree with ToMi's labels depending on the noun. Table 4.1 presents an example where the object and container have a large effect on human responses.[5]

Resolving Unintentional Ambiguities ToMi's story construction process often leaves object locations ambiguous, which forces humans to (incorrectly) rely on their physical commonsense. For example, the location of the *basket* in line 4 of Table 4.1 is ambiguous.

[5]Using Amazon Mechanical Turk, we present 20 humans with the template in Table 1, using either *(hat,box)* or *(apple, pantry)*. Workers are paid $1 per HIT.

This ambiguity is at times resolved at a later step in the story (Arodi and Cheung, 2021), but it is not true for all cases, and these resolutions were not expressly intended by ToMi's original design. This complicates the task beyond theory of mind. For example, in Table 4.1, the reader must conclude from *"Oliver is in front yard"*, *"Oliver moved the objectA (...)"*, and *"The objectA is in basket"* that the basket is in the front yard, and hence that Ethan saw it there. This requires 3-hop reasoning, and knowing ahead of time that, in ToMi, characters do not change rooms unless explicitly stated.

To solve these unintentional ambiguities and additional 3-hop reasoning requirements, and instead solely measure theory of mind reasoning skills, we automatically add a sentence that disambiguates the location of each container immediately after each primitive (c) or (e) (e.g., adding *"The basket is in the front yard"* as line 5 in Table 4.1). Finally, as reported in Arodi and Cheung (2021) and Sap et al. (2022), ToMi contains some mislabeled second-order questions, which we also correct.

4.5 Experiments

We experiment with several base LMs, and evaluate each of them both out-of-the-box via zero-shot prompting, and by applying SymbolicToM to ToMi stories to produce answers. We evaluate Macaw-3B (Tafjord and P. Clark, 2021), GPT3-{Curie,Davinci} (Brown et al., 2020), Flan-T5-{XL,XXL} (Chung et al., 2024), LLaMA-{7B, 13B} (Touvron et al., 2023a), GPT3.5 (OpenAI, 2022), and GPT4 (OpenAI, 2023). We use WANLI (A. Liu et al., 2022) for identifying NLI contradictions, and the AllenNLP library (Gardner et al., 2018) for OpenIE. We additionally refine each subject and object in extracted triples to remove any stopwords that may be accidentally included by OpenIE.

We first evaluate SymbolicToM's performance as a plug-and-play method for different base LMs on ToMi (§4.5.1). We then test whether performance gains are robust to ToMi story structure modifications (§4.5.2). Finally, we explore SymbolicToM's robustness to linguistic diversity (§4.5.3).

Supervised Models For comparison, we train two supervised models: Textual Time Travel (TTT) (Arodi and Cheung, 2021), and a fine-tuned GPT3-Curie. TTT is a modification of EntNet (Henaff et al., 2017) designed for theory of mind tasks; GPT3-Curie is finetuned on 6000 ToMi examples for one epoch. GPT3-Curie achieves near-perfect performance when finetuned on ToMi (98.5% accuracy when averaging all questions; Table C.3). Interestingly, GPT3-Curie achieves a higher accuracy than the theory of mind-motivated TTT (accuracy 92.3%). We explore model robustness in §4.5.2.

4.5.1 In-Domain Evaluation

We evaluate all base LMs comparing their performance out-of-the-box, versus when adding SymbolicToM. Figure 4.4 shows results by question type, showing dramatic improvements for all theory of mind questions: +62 points in accuracy for first-order false-belief questions for Flan-T5-XL, +78 points in accuracy for second-order false-belief questions for GPT3.5,

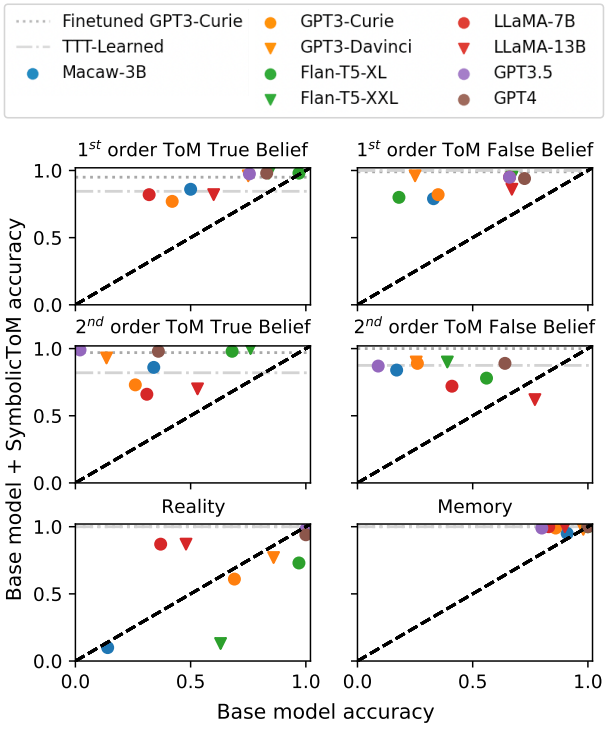


Figure 4.4: Accuracy for each ToMi question type and base model (higher is better). Dots in the upper triangle have higher performance with SymbolicToM than the base model out-of-the-box. Horizontal lines give supervised models' performance. Full results in Table C.3.

among other improvements. In addition, we observe all models maintain near-perfect performance with and without SymbolicToM in memory questions. Supervised models show high accuracy for all question types.

We only see significant decreases in performance for reality questions in Flan-T5 models. This can be partially attributed to the questions' phrasing: questions are posed as "Where is the celery *really*?". Removing *really* results in 96% accuracy for Flan-T5-XL. Flan-T5-XXL empirically shows a bias towards providing a room rather than container as an answer when only one container is mentioned, which is often the case for SymbolicToM-filtered stories. Rooms are invalid answers in ToMi. An ablation on the final filter function of Algorithm 2 suggests that keeping more containers in the final story reduces this bias and still yields significant improvements for false-belief questions across all models (see §C.3.1). Besides *reality* questions, Flan-T5-XXL with SymbolicToM achieves results comparable to the supervised TTT.

4.5.2 Story Structure Robustness Test Sets

We create three test sets by modifying ToMi's stories structures without adding new types of actions or linguistic diversity. These tests were only evaluated once, after finishing development of SymbolicToM. Test sets are defined below. See Appendix C.2.2 for concrete examples.

	D_1	D_2	D_3
Off-the-shelf models			
Macaw-3B	8	12	30
Flan-T5-XL	86	51	68
Flan-T5-XXL	69	59	52
GPT3-Curie	37	39	57
GPT3-Davinci	20	25	39
GPT3.5[6]	1	0	48
GPT4	58	62	97
LLaMA-7B	17	17	17
LLaMA-13B	26	36	37
SymbolicToM + Off-the-shelf models			
Macaw-3B	89 (+81)	71 (+60)	70 (+41)
Flan-T5-XL	76 (-10)	96 (+46)	**100** (+33)
Flan-T5-XXL	93 (+24)	**100** (+41)	**100** (+49)
GPT3-Curie	84 (+48)	81 (+42)	73 (+16)
GPT3-Davinci	92 (+73)	91 (+66)	90 (+50)
GPT3.5	**100 (+99)**	**100 (+99)**	99 (+51)
GPT4	**100** (+42)	**100** (+38)	**100** (+4)
LLaMA-7B	99 (+82)	92 (+75)	88 **(+71)**
LLaMA-13B	78 (+52)	84 (+48)	84 (+47)
Supervised models			
TTT	49	65	78
Finetuned GPT3	51	68	32

Table 4.2: Precision using SymbolicToM on all questions from 100 stories for each of the modified test sets D_i. Supervised models were trained on ToMi; all others do not require training. Parenthesis reflect differences between using and not using SymbolicToM: **bold** reflects higher overall performance, and **green** reflects the highest net improvements when using SymbolicToM.

Double Room False Belief Story (D_1) Two false belief substories involving the same two characters p_1, p_2 are concatenated to yield a longer, more complex story. Each substory has different objects being moved, across different containers. The system is probed using all four combinations of second-order theory of mind questions involving the two characters and locations. Questions are evenly split between the first and second substory.

Three Active Characters Story (D_2) Three characters p_1, p_2, p_3 are in the same room, where an object o_1 and three containers c_1, c_2, c_3 are available. The story is as follows: p_2 leaves before p_1 moves o_1 from c_1 to c_2, but p_3 witnesses the move. Then, p_1 leaves the room. Later, p_3 moves the object to container c_3 without any witnesses. The system is probed using all combinations of second-order theory of mind questions.

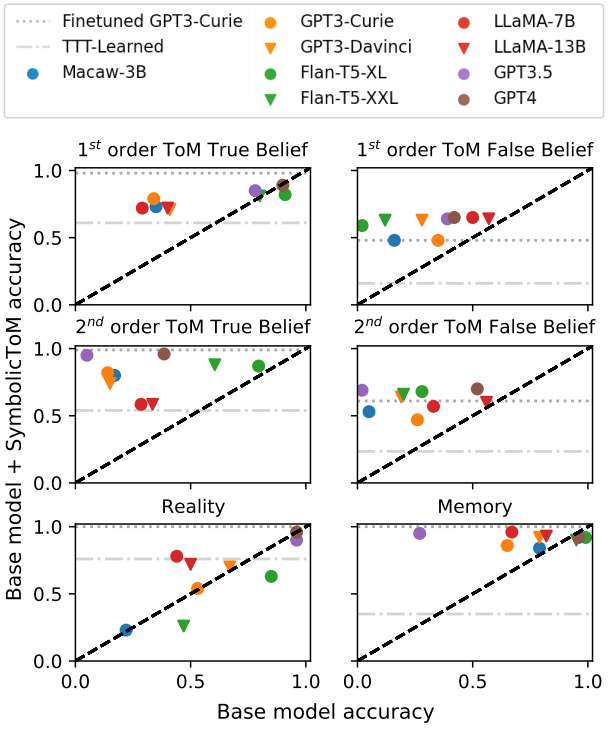


Figure 4.5: Results for ParaphrasedToMi when prompting GPT3 as implementation of RESULTINGSTATE (Davinci for all except for Curie). Dots in the upper triangle imply performance with SYMBOLICTOM is higher than using the base model out-of-the-box. Horizontal lines reflect supervised models' performance (higher is better).

Multiple Object Movements Across Four Containers (D_3) Two characters p_1, p_2 are in a room, with a single object, and four containers $c_1, \ldots, c_4$. p_1 moves the object from c_1 to c_2 and right before leaving the room, p_2 enters. p_2 then moves the object to c_3, and then c_4. We probe with all first and second-order theory of mind questions.

Results Supervised models significantly overfit to ToMi's original story structures (Table 4.2). In contrast, all models had high accuracy when equipped with SYMBOLICTOM, especially larger models, such as GPT3.5, LLaMA-{7B,13B}, among others.

D_2 may also be used to test third-order ToM reasoning, asking questions such as "Where does p_1 think that p_2 thinks that p_1 will search for the o_1?". Third-order ToM is a reasoning depth currently untested by available NLP benchmarks. SYMBOLICTOM consistently enhances the performance of off-the-shelf LLMs and outperforms supervised methods in the third-order ToM setting. See details in Appendix C.3.2. This experiment showcases how extensions of ToMi may be used to test higher-order reasoning. This is the first approach towards testing third-order ToM in LLMs; a benchmark to comprehensively test such order of reasoning exceeds the scope of this paper.

4.5.3 Paraphrasing Robustness Evaluation

We assess the robustness of all models when utilizing various wordings for each sentence. We reword all templates using GPT3-Davinci, utilizing different choices of objects, rooms, and names, and manually excluded incorrect paraphrases. The resulting dataset—ParaphrasedToMi—exhibits much greater complexity, as these rewordings can express actions in a less straightforward way. All paraphrases are shown in Appendix C.2.1.

Figure 4.5 demonstrates significant performance decreases for supervised models transferring to ParaphrasedToMi. TTT's average accuracy drops 54 points from ToMi, with losses across all question types. Finetuned GPT3 exhibits significant losses in false-belief questions (-40 average accuracy) but is robust for other question types.

Methods without supervision also suffer significant losses, but SymbolicToM still results in large improvements for theory of mind questions. Models equipped with SymbolicToM perform significantly better than the supervised TTT model across all theory of mind questions. ParaphrasedToMi is significantly more difficult for SymbolicToM since it triggers more errors in edge removal (due to errors in NLI classification), as well as errors in edge insertion (due to errors in the resulting state's triple extraction). Although computing resultingState by prompting the base LMs was successful with original phrasings (as defined in §4.3.2.1), we observed differences in robustness when prompting with paraphrases. We found implementing resultingState with GPT3 reliable, and thus we use it for all models. Results using other models are included in §C.3.3: false-belief performance is even better for models like LLaMA, GPT3.5, or GPT4.

4.6 Related Work

Existing Approaches Classical reasoning tasks require achieving some goal, e.g., proving a statement, given a set of facts and universally valid rules (e.g., Tafjord, Dalvi, and P. Clark, 2021). A common approach is to decompose the target reasoning task into subtasks, for example by using off-the-shelf LMs (Creswell, Shanahan, and Higgins, 2023; Kazemi et al., 2023; Nye et al., 2021). We use a similar technique in SymbolicToM, breaking the higher-level reasoning task into graph reasoning subtasks. Nonetheless, these approaches cannot be simply ported to our domain: stories' facts (i.e. the world state) change over time and are not universally accessible to all characters, and commonsense rules and assumptions like object permanence must made explicit. SymbolicToM's design addresses these challenges by maintaining and updating graphs about facts and beliefs as a story progresses.

In scenarios where world state changes over time, such as in text-based games, existing approaches maintain and update structured world representations as the world state changes (Adhikari et al., 2020; Ammanabrolu and Riedl, 2021). However, while these approaches could potentially be applied in our scenario to update G, they would not address the problems of multiple-belief representation or knowledge propagation to witnesses' graphs, with some approaches even being explicitly impossible for modeling second-order ToM (Qiu et al., 2022).

ToM beyond NLP Theory of mind is also crucial in multi-agent reinforcement learning (Rabinowitz et al., 2018), including in bidirectional symbolic-communication (Sclar, Neubig, and Bisk, 2022; Y. Wang et al., 2022b), unidirectional natural-language settings (Zhu, Neubig, and Bisk, 2021); and recently, by combining reinforcement learning, planning, and language, to create a human-level Diplomacy player ((FAIR) et al., 2022). It has also received increased attention in human-computer interaction (Q. Wang et al., 2021) and explainable AI (Akula et al., 2022).

Psychologists divide theory of mind into two types of reasoning: affective (emotions, desires) and cognitive (beliefs, knowledge) (Shamay-Tsoory et al., 2010), with the former developing earlier in children (Wellman, 2014). Our work focuses on the latter, but the principle of multiple belief representation could also be applied to affective theory of mind reasoning. Existing work has shown that humans are proficient at second-order or higher false-belief reasoning, also referred to as *advanced ToM* (Białecka-Pikul, Kołodziejczyk, and Bosacki, 2017), with evidence that we can perform even third- and fourth-order reasoning (Osterhaus, Koerber, and Sodian, 2016; Valle et al., 2015). While, to best of our knowledge, no dataset requires beyond second-order ToM, SymbolicToM explicitly models the recursive reasoning that supports queries of any reasoning order.

4.7 Conclusions

Theory of mind is an essential social intelligence ability. Developing agents with theory of mind is requisite for a wide range of applications, including reading comprehension, tutoring, dialogue, personalization, and negotiation. For example, in reading comprehension settings (and broadly for natural language understanding), having a multi-level understanding of texts is crucial for providing meaningful and contextualized answers: stories often rely on theory of mind reasoning to create conflict (e.g., in murder mysteries, drama, and romances, as in the final acts of *Romeo and Juliet*).

We present SymbolicToM, a plug-and-play method to enable theory of mind reasoning in language models via explicit symbolic representations in the form of nested belief states. SymbolicToM requires no training or fine-tuning, a key aspect for a domain with scarce supervised data and limited success in learning from massive unlabeled text alone. With experiments on reading comprehension tasks, our approach demonstrates dramatic improvement in the accuracy of base language models, especially for false-belief scenarios.

We also show that, in contrast to supervised methods, SymbolicToM is highly robust to story perturbations and out-of-domain inputs where supervised methods suffer significant degradations (as in, e.g., P. Yu et al., 2023).[7] Our results show the promise of augmenting neural language models with symbolic knowledge for improving their social reasoning skills. We leave to future work to investigate similar approaches for other types of social intelligence; as well as develop new datasets that cover a more diverse set of interactions.

[7] As a part of out-of-domain testing, we also create a more challenging version of the available ToM datasets, available at https://github.com/msclar/symbolictom along with a corrected version of ToMi.

Limitations

SymbolicToM assumes stories are written chronologically, which may not hold for some human-written stories. This may be alleviated using time-stamping models like Faghihi and Kordjamshidi (2021). Furthermore, since we use off-the-shelf models (WANLI (A. Liu et al., 2022) and OpenIE (Stanovsky et al., 2018)) to create and update the graphs, the presented approach may propagate errors as revealed in the linguistic diversity experiments. However, these issues can be largely alleviated by using more sophisticated models, even the LLMs like GPT3 themselves. We do not experiment with them due to budgetary restrictions.

Currently, all NLP datasets available for theory of mind reasoning describe Sally-Anne tests. In these datasets, the concept of large distances is absent, meaning that anyone specified to be in a location is assumed to be a witness of the actions that occur there. This assumption can be violated in realistic settings. For example, *"Anne is in the USA"* does not imply she is a witness to every action happening in the USA. In future work, this approach can be improved by refining the witnesses detection algorithm to incorporate physical commonsense reasoning. We could also refine the witness detection algorithm by sampling paths between the inserted edge and each node referring to a person, to query an LM directly on that substory by asking if the person witnessed the action. To be able to test both of these ideas, we would need to obtain new theory of mind datasets with significantly more types of interactions and physical commonsense in the stories.

Chapter 5

Program-Guided Adversarial Data Generation for Theory of Mind Reasoning

*This chapter is based on **ExploreToM**: Explore Theory of Mind: Program-Guided Adversarial Data Generation for Theory of Mind Reasoning. Published at ICLR 2025.*

5.1 Introduction

While Chapter 4 showed that inference-time structure can substantially improve theory of mind reasoning by compensating for models' poor performance at test time, such approaches carry additional computational cost and their benefits cannot be readily transferred to downstream applications that may require their own customized inference-time algorithms. This motivates an alternative pathway: directly addressing the data bottleneck discussed in §3.9 to improve the underlying model while simultaneously enabling automatic stress testing of models' theory of mind capabilities.

Concretely, we introduce **ExploreToM**, an A*-powered algorithm for generating reliable, diverse, and challenging theory of mind data that can be effectively employed for fine-tuning or testing LLMs. Our approach leverages a domain-specific language to generate synthetic story structures and their characters' mental states. We then use LLMs to create plausible stories based on these plots, allowing for precise control over the narrative and tracking each character's mental state with high confidence. We employ A* search (Hart, Nilsson, and Raphael, 1968) to efficiently navigate the vast space of possible narratives and pinpoint those that are most likely to fool state-of-the-art LLMs. This in turn allows us to create a robust, rich dataset that effectively tests the limits of current models (Fig. 5.1). By generating story structures separately from lexical realizations, we can distinguish the model's core understanding of social reasoning from vocabulary cues that might give away stylistic hints.

ExploreToM's contributions are three-fold, addressing both empirical challenges identified in §3.9 in tandem: we algorithmically address blind spots in theory of mind evaluation, we provide a recipe to create complex training data that helps imbue models with better

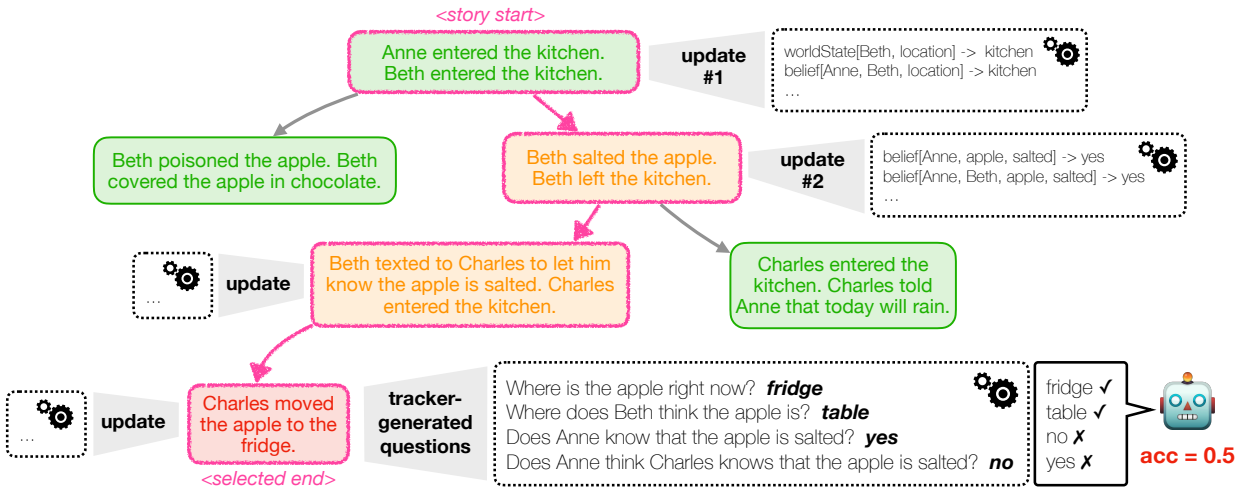


Figure 5.1: ExploreToM finds challenging stories for a given language model by searching through the space of stories supported by its domain-specific language for mental state tracking (⚙), sampling k supported actions at a time (shown as a node, $k = 2$ in the example). Difficulty evaluation (simplified in the figure as easy, medium, hard) of each partial story is done through automatically generated questions with reliable ground-truth answers thanks to our tracking procedure.

theory of mind reasoning skills, and we provide insights into why theory of mind skills are still elusive for LLMs.

First, our work helps to address the evaluation challenges discussed in §3.9: conflicting results in the literature, limited benchmark coverage, and saturation risk. ExploreToM's adversarial nature allows for automatically generating stories to stress test any LLM, diminishing the risk of data leakage onto training data, and thus being more robust than manually-crafted benchmarks. Our experiments show that **ExploreToM-generated data is extremely challenging, with GPT-4o and Llama-3.1 70B accuracies as low as 9% and 0% respectively.** ExploreToM supports significantly more scenarios than previously possible, with the unique addition of knowledge gain asymmetry during an interaction, among other improvements.

Second, our method allows creating complex and diverse theory of mind data that can be leveraged for model fine-tuning. As shown in Section 4.5.2, existing data sources are not suitable for this purpose: fine-tuning on them produced models that overfit to specific story structures rather than learning the underlying reasoning. Instead of continuing the inference-time line of work shown in Chapter 4 and in subsequent efforts (Jung et al., 2024; Wilf et al., 2024; P. Zhou et al., 2023), ExploreToM makes supervision viable where it was not before. Fine-tuning Llama-3.1 8B Instruct on ExploreToM's data achieves a substantial **+27 accuracy point improvement on the classic ToMi benchmark** (Le, Boureau, and Nickel, 2019), **showing good generalization to even more complex ExploreToM stories than those seen during training, while still retaining general reasoning capabilities**.

Finally, ExploreToM enables providing new insights into why basic theory of mind

reasoning is still challenging for LLMs. We show that LLMs struggle with basic state tracking, a fundamental skill underlying theory of mind reasoning: tracking mental states necessarily requires being able to track states. Our experiments also reveal that in order to improve on theory of mind during fine-tuning, it is crucial to use data that requires theory of mind as opposed to simply requiring state tracking. However, found data (either in-the-wild, or randomly generated) is unlikely to have this necessary property, which may be a key contributor to lagging model performance. Overall, ExploreToM offers a valuable tool for advancing theory of mind research, enabling the development of more effective LLMs that can better handle complex social interactions.

5.2 Adversarially constructed stories with ExploreToM

Building on the standard approach in theory of mind of assessing mental state understanding through question answering (Baron-Cohen et al., 1999; Kinderman, Dunbar, and Bentall, 1998; Wimmer and Perner, 1983), ExploreToM creates stories where different characters may have different beliefs about the current world state and about other people's beliefs, paired with questions to probe model understanding (see Fig. 5.1's highlighted story, along with associated questions probing understanding that e.g. "Anne does not know that Charles knows that the apple has been salted").

ExploreToM's story generation process is divided into three main steps: plausible story context sampling (Section 5.2.1), adversarial story structure generation (Section 5.2.2), and optionally story infilling (Section 5.2.3) – an example is outlined in Figure 5.2. We automatically generate questions to probe understanding of said stories as part of the adversarial story structure generation process (Section 5.2.2.2); this process finds challenging story structures, i.e., story structures that would yield low accuracy with our generated questions. Because questions are generated automatically and directly from the tracked mental and world states, ground truth answers have a high degree of reliability: we do not use language models at all in the question-answer generation procedure.

5.2.1 Plausible story context sampling

We use an LLM zero-shot to generate a consistent and plausible story context, comprising essential elements such as character names, roles, locations, relevant objects, object containers, and discussion topics (see Fig. 5.2A for a full example). This single-step process ensures a coherent and believable setup for our theory of mind stories. Previous approaches (such as ToMi (Le, Boureau, and Nickel, 2019)) sample objects (e.g., an apple) and object containers independently (e.g. a bottle), often resulting in commonsense violations. Unlike these approaches, our method generates a coherent context by sampling these elements jointly in a single LLM call: autorregresive LLMs will naturally suggest contextually plausible elements based on the ones they already generated, and especially so when explicitly requesting it in the prompt. Additionally, we sample possible object state updates (Figure 5.2B), which are then refined through using an LLM as a judge to filter out implausible and low quality generations. The role of these state updates will be discussed in further detail in Section 5.2.2.1. The exact prompts used for sampling story contexts are shown in App. D.4.

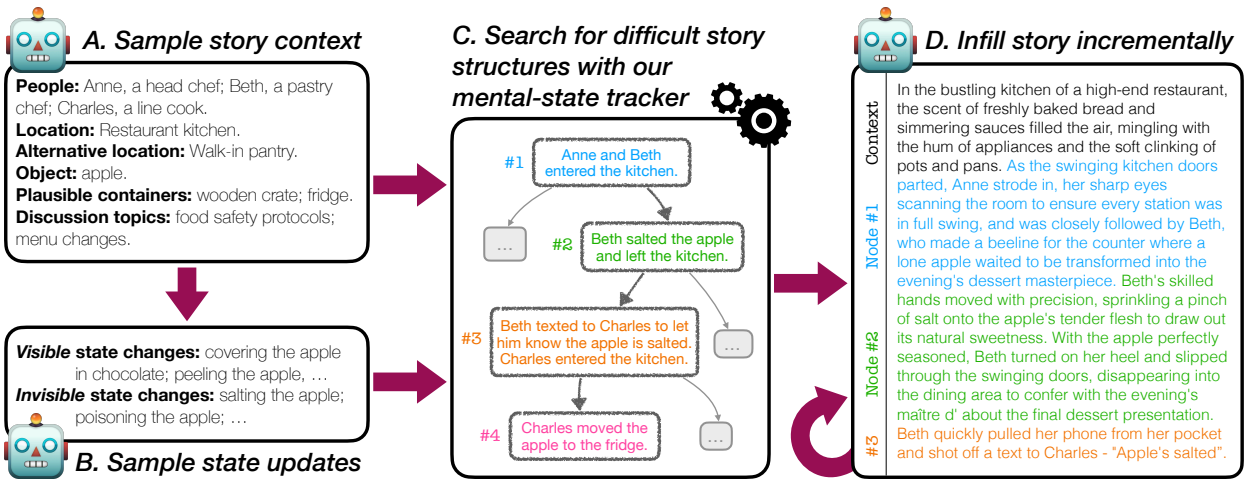


Figure 5.2: Overview of ExploreToM's story generation procedure. We first sample a plausible story context using an LLM (shown in A and B). Topics discussed, location changes of objects and people, and object state updates, may all be required to track in order to pass our theory of mind tests. We then search for difficult story structures (i.e., the raw story points) by sampling and analyzing different orders in which these actions may be performed using A* search (shown in C, and Fig. 5.1). This ensures that the resulting stories will all be challenging tests for models, and may be used for further improvement. Finally, these story structures (nodes #1-4) are iteratively infilled, one story action at a time, using a language model, yielding a natural-sounding story. Infilled stories are used as training data; benchmarking is done with story structures since they have the highest reliability.

5.2.2 Adversarially Generating Challenging yet Plausible Story Scripts

5.2.2.1 Theory of Mind-Specific Language Definition

ExploreToM's theory of mind-specific language consists of a diverse set of actions $\mathcal{A}$, each transforming the world state and the beliefs of the people involved (the story *state* $s \in \mathcal{S}$). A *story* is thus defined as a sequence of actions $(a_1, \ldots, a_n)$, where each action $a_i \in \mathcal{A}$ is a function $a_i : \mathcal{S} \to \mathcal{S}$. Each action also has preconditions to be able to apply it, i.e., restrictions to its domain. For example, a precondition for "Charles entering the kitchen" is to not be in it already. Applying an action also automatically updates our world state tracking and belief tracking: for example, "Charles is now in the kitchen"; "Anne knows that Charles is in the kitchen since they were also in the kitchen"; "Charles knows that Anne is in the kitchen since he can see her"; and so forth. All these updates and conditions are specifically programmed and tested; see App. D.1 for the full programs.

ExploreToM enables the generation of diverse stories by significantly expanding the range of supported actions. These actions include physical changes to the world state such as entering and leaving a room (denoted a_{enter}, a_{leave}), moving an object to a container (or in general, updating its state; denoted $a_{\text{moveObjContainer}}$, $a_{\text{updateObjState}}$ respectively), relocating an object to a different room ($a_{\text{moveObjRoom}}$). Additionally, ExploreToM supports various forms of communication, including: private conversations between two characters, or public

broadcasts to all characters in a room; casual discussions about a topic (denoted *chit-chat*), or notifications about changes in the world state (denoted *info*); these actions are referred to as $a_{\text{info-private}}$, $a_{\text{info-public}}$, $a_{\text{chitChat-private}}$, and $a_{\text{chitChat-public}}$. These actions can occur at any point in the story, allowing for a rich and dynamic narrative (see formal definition in App. D.1) and expanding prior work (Wu et al., 2023).

Each new action requires carefully writing the implied belief and world state updates, which precludes scaling the number of actions supported. However, we alleviate this by noting that from a theory of mind perspective, many actions are equivalent. For example, "peeling an apple" or "covering an apple in chocolate" have the same implications with respect to belief updates (a *visible property* of the apple is being updated, and the witnesses would be the same). Similarly, poisoning an apple has the same implications as moving an apple from a drawer to a fridge (an *invisible property* is updated, witnesses would be the same, and non-witnesses would not assume there has been an update). The instantiations of these equivalent state updates from a belief perspective are done with an LLM during the story context sampling (see Figure 5.2.B).

Asymmetric belief updates In prior work, all belief updates were *symmetric*: if A and B witnessed an action, then A knows that B witnessed the action and vice versa. Our framework introduces the ability to model asymmetric scenarios. Specifically, we enable the addition of secret witnesses to an action such as someone observing through a security camera, or removal of witnesses without others' knowledge, as in the case of someone becoming distracted by their phone. This added nuance allows for more realistic and complex social scenarios. Asymmetries a_{peek} and $a_{\text{distracted}}$ are modifier functions, e.g., as a modifier to "Beth salted the apple" ($a_{\text{updateObjState}}(\cdot)$) there may be a secret person peeking ($a_{\text{peek}}(a_{\text{updateObjState}}(\cdot))$): "While this was happening, Diane witnessed it in secret."

5.2.2.2 Generating Questions and Assessing Resulting Story Difficulty

We assess a model's understanding of a generated story $s{=}(a_1, \ldots, a_n)$ by probing it with automatically generated question-answer pairs. ExploreToM-generated answers are more reliable than purely-LLM generated ones, since they are directly produced from the states' trajectory with our tracker. Questions may be testing first-order beliefs, second-order beliefs, or regular state tracking: *First-order* refers to asking about someone's mental state (e.g., "Does Anne know the apple is salted?"); *Second-order* refers to one extra level of recursion in mental state tracking (e.g., "Does Anne think that Charles know the apple is salted?"); *State tracking* may probe about the current state (*ground truth*) or prior ones (*memory*).

We expand the complexity of memory questions with respect to prior work by asking about any intermediate state (e.g. "Where was the object before X happened?") instead of solely about the initial one ("Where was the object at the beginning?"). Our generated questions are simple to evaluate: they are either binary (yes/no), or are answered by stating an object, container, or room. Specific question formulations differ based on the property, e.g., location ("Where does Charles think that Anne will search for the apple?") or knowledge ("Does Charles know that the apple is salted?"). See App. D.2 for the full list of supported questions.

A question is considered *interesting* if the answer would change depending on the person

being asked about. For example "Does Anne think that Charles knows that the apple is salted?" is interesting because the answer would differ if asked about someone else, such as "Does Beth think that Charles knows the apple is salted?". ExploreToM's tracker very easily allows for automatically detecting interestingness.

5.2.2.3 A* Search

Given a context $\mathcal{C}$ and a set of actions $\mathcal{A}$, our main goal is to find challenging story structures. To increase ExploreToM's usage flexibility, we support the option of searching for stories s that fulfill desired user conditions $\textbf{isDesired}(s) \in \{0, 1\}$, such as the number of people involved, or the number of actions belonging to a subset $\mathcal{A}' \subseteq \mathcal{A}$ of *important actions*.

We search over the space of plausible story structures of up to m actions. We define this space as a directed graph, where each node is a sequence of valid actions $s = (a_1, \ldots, a_i)$, and there is an edge between s and s' if and only if s is prefix of s', and s' contains k more actions than s. $k \geq 1$ is the *grouping factor* for actions, defining the granularity with which we will sample and evaluate nodes. For simplicity, Figure 5.1 depicts only the new $k = 2$ actions that each node introduces.

To find challenging stories that simultaneously fulfill the user constraints we use A* search (Hart, Nilsson, and Raphael, 1968). By definition, A* selects the path that minimizes $f(s) = g(s) + h(s)$, where $g(s)$ is the cost of the path from the start to node s, and $h(s)$ is a heuristic that estimates the cost of the cheapest path from s to a *goal node* (one of the nodes where it would be acceptable to finish the search). In our context, goal nodes are those such that $\textbf{isDesired}(s') = 1$. We choose A* as our search algorithm precisely because it enables to search this space prioritizing desired user conditions through $h(s)$, as we will detail below.

A story is said to be challenging for a model if it incorrectly answers our generated questions, i.e., it shows low accuracy. Thus, we define $g(s)$ as our target model's accuracy among *all* questions for s. We define the heuristic function $h(s)$ as a proxy estimation of the likelihood of generating a full story $s + s'$ that fulfills user constraints $\textbf{isDesired}(s) = 1$, where s' is the continuation of story s:

$$h(s) = \alpha\left(1 - \frac{1}{P}\sum_{i=1}^{P} \mathbb{1}(\textbf{isDesired}(s + s'_i) = 1)\right)$$

Here, all s'_i are randomly sampled continuations of s and $0 \leq \alpha \leq 1$ is a scaling factor. A* requires to evaluate all neighbors of a node s. Since this would be infeasible given the vast space to explore, and that each $f(\cdot)$ evaluation requires several LLM calls (one per question), we restrict the evaluation to a pre-defined constant number of neighbors, prioritized by the closeness of this node to fulfilling the conditions described by $\textbf{isDesired}(\cdot)$. This pre-defined constant may depend on $f(s)$ to prioritize more promising partial stories (i.e., with lower $f(s)$ values).

5.2.3 Story infilling

Story infilling is the process of transforming a full story structure $s = (a_1, a_2, \ldots, a_n)$ with a story context $\mathcal{C}$ into a natural-sounding narration (see Fig. 5.2D). We infill stories iteratively

with an LLM by transforming each action a into a more natural sounding one, according to some stylistic desiderata d, and conditioned on the previously infilled context z (denoted infill(a, z, d)). Supported stylistic desiderata d are length requests (e.g., "use up to two sentences") or style requests (e.g., "make this into a conversation"); we optionally also include sampled character goals g and an initial narration context c based on the story s, also generated with an LLM (e.g., Anne's goal may be to oversee that all dishes are rapidly delivered to customers; see initial context example in Fig. 5.2). Concretely, the full story infilling SI is as follows:

$$SI(i) = \text{infill}(a_i, SI(i-1), d_i, g) \text{ where } SI(0) = c$$

Infilling is done iteratively to ensure that the order of the actions stays the same, since this is important for keeping the mental state tracking valid. To further increase reliability, we use an LLM as a judge after each infilling step to confirm that each mental state tracked after executing the story step a_i still holds even after infilling. This discards infillings that introduced ambiguity or hallucinations.

5.3 ExploreToM as an evaluation benchmark

We begin by showcasing how ExploreToM story structures can be used as a challenging benchmark, highlighting its unique features and advantages.

Experimental setup We use ExploreToM to generate 10 story structures for each of 9 action sets (each with and without asymmetry) and each set of user conditions. Each story generation is allowed to evaluate 50 nodes. User conditions—isDesired$(\cdot)$—require exactly $p \in \{2, 3, 4\}$ people involved, with $a \in \{2, 3, 4\}$ actions belonging to the set of important actions $\mathcal{A}'$, spanning across either $r = 1$ or $r = 2$ rooms, and with $m \leq 15$ actions in total—leading to a total of 162 settings. In all experiments, $\mathcal{A}'$ are the actions that add new basic world knowledge: $\mathcal{A}' = \{a_{\text{moveObjContainer}}, a_{\text{updateObjState}}, a_{\text{moveObjRoom}}, a_{\text{chitChat-*}}\}$. We then infill every story. We use Llama-3.1-70B-Instruct (Dubey et al., 2024), GPT-4o ((OpenAI, 2024); queried early Dec. 2024), and Mixtral-8x7B-Instruct (A. Q. Jiang et al., 2024) to generate story structures. A* is run with $\alpha = 0.1$, $P = 50$, and $k = 3$ (i.e. grouping three actions per node). See generation examples in App. D.5.

ExploreToM finds challenging story structures for frontier models As shown in Table 5.1, our ExploreToM consistently identifies story structures that are highly challenging for models across various action sets, with average performances in ExploreToM-generated datasets as low as 0.09 for GPT-4o (i.e., 9%). When increasing the number of actions, difficulty tends to increase or remain similarly challenging. Performance tends to stay the same or decrease when increasing the number of people involved, possibly because with a fixed number of state-changing actions there will be fewer actions per person which may be easier to track. See Figure 5.3 and App D.3.5.

Table 5.1: Accuracy results of ExploreToM's story structures on 18 action sets $\mathcal{A}$, each aggregating 90 total stories from 9 different settings (number of people, actions, and rooms). Each set is either based on actions supported by well-known theory of mind tests or includes our novel expansions, and is analyzed excluding or including asymmetry (✗, ✓). Each setting requires at least one action in the story to be from one of the $\boxed{\text{squared}}$ actions to encourage non-overlapping story structure characteristics between action sets shown. Data was generated using each model as its own evaluator (i.e., as $g(\cdot)$), and results shown include all first-order questions—the most basic theory of mind level, not requiring recursion. Lowest accuracy for each model is bolded.

ExploreToM action set $\{a_{\text{enter}}, a_{\text{leave}}, \ldots$	Llama-3.1 70B Inst.		GPT-4o		Mixtral 8x7B Inst.	
include asymmetry modifiers? (a_{peek}, $a_{\text{distracted}}$)	✗	✓	✗	✓	✗	✓
$\ldots, \boxed{a_{\text{moveObjContainer}}}\}$	.18	**.00**	.40	.25	.37	.33
$\ldots, \boxed{a_{\text{updateObjState}}}\}$	.27	.24	.25	.24	.03	.01
$\ldots, \boxed{a_{\text{moveObjContainer}}}, \boxed{a_{\text{updateObjState}}}\}$	.26	.03	.35	.31	.24	.09
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{moveObjRoom}}}\}$	.11	.10	**.09**	**.16**	**.00**	**.00**
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{info-*}}}\}$	**.06**	.07	.29	.25	.35	.36
$\ldots, a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}, \boxed{a_{\text{info-*}}}\}$	.11	.07	.24	.24	.03	.04
$\ldots, a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}, a_{\text{chitChat-*}}, \boxed{a_{\text{info-*}}}\}$	.72	.69	.73	.68	.53	.47
$\ldots, \boxed{a_{\text{chitChat-private}}}\}$	.75	.66	.77	.59	.51	.45
$\ldots, \boxed{a_{\text{chitChat-public}}}\}$	.60	.55	.49	.47	.37	.34

A* is a better strategy than over-generation and filtering Over-generation and filtering has become a standard procedure for synthetic data generation (e.g. Y. Wang et al., 2023; West et al., 2022). We measure the effectiveness of A* by comparing the A*-generated data to the data resulting from over-generating stories with our domain-specific language—using the same **isDesired**$(\cdot)$ criteria and budget as used in the A* search—and retaining only the most difficult stories. In a set of 81 randomly-selected settings (50% of the original 162 settings, due to the experiment's high cost), we generate 50 stories with each method using Llama-3.1-70B-Instruct and a budget of 2500 accuracy evaluations each. A* yielded a more challenging dataset (by 2 accuracy points), with shorter stories on average (1.6 fewer actions). This length difference is possibly due to the pressures A* induces towards shorter stories through the heuristic $h(s)$. See Figure D.1 for the full distribution of results.

Story structures found adversarially for a model remain challenging for other models We evaluate the difficulty of a ExploreToM-generated dataset with each model, and find that although there is an increased difficulty towards data generated adversarially with the same model, it remains challenging for all others. Notably, the generated datasets remain challenging even when adding question types not included in the $g(\cdot)$ optimization (second-order belief questions). See Table 5.2.

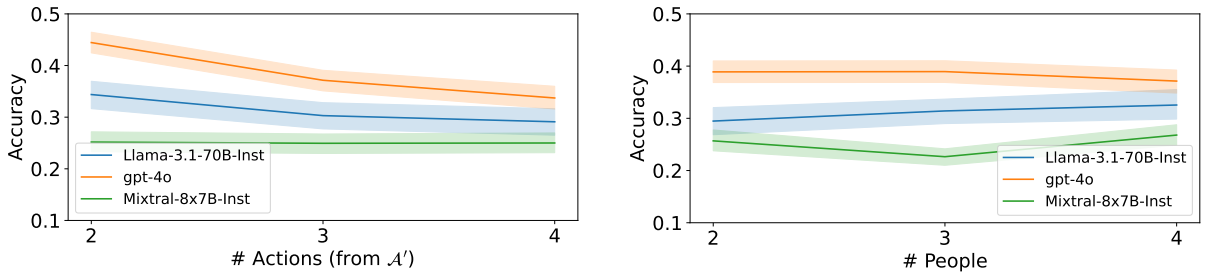


Figure 5.3: Accuracy on ExploreToM's story structures when increasing the number of actions or people involved. Accuracy is computed across all story structure settings. Difficulty of ExploreToM-generated stories tends to increase or stay similar when increasing the number of actions. A story with greater number of people suggests similar or lower difficulty, possibly because when fixing the number of actions there are fewer actions per person (see details in App. D.3.5).

Table 5.2: Accuracy results on ExploreToM-generated data built to minimize accuracy for each particular model. A random sample of 1000 (story structure, question) pairs is shown. Data remains challenging even if it was built with a different model, and even including questions we did not optimize for: story structures were selected adversarially towards first-order belief questions only ($g(\cdot)$), accuracies shown include all belief questions.

	Model used for evaluation		
Model used in ExploreToM generation ($g(\cdot)$)	Llama-3.1 70B Inst.	GPT-4o	Mixtral 7x8B Inst.
Llama-3.1 70B Inst.	0.57	0.68	0.32
GPT-4o	0.60	0.61	0.32
Mixtral 7x8B Inst.	0.68	0.74	0.30

Humans agree with ExploreToM-generated story structures labels We conducted a human evaluation to verify the quality of the story structures' automatically-generated labels and the story infillings. For labels, we annotated 100 questions across 12 randomly-sampled story structures from all settings generated for Table 5.1, and found 99% agreement with our expected answers—likely due to the clear and concise nature of our stories and that the ground truth labels were generated by our domain-specific language. We measure story infilling quality by repeating the question-answering procedure with a different set of 100 questions across 12 randomly-sampled infilled story structures. In this case, the human agreed with the ground truth label 89% of the time—a small degradation likely due to the LLM-powered method introducing ambiguity.

Infilled stories remain challenging Infilled stories with Llama-3.1 70B yielded an average accuracy of 0.61. Although the average accuracy increased by 0.12 through the infilling process, the samples remained challenging thanks to the highly challenging underlying sto-

Table 5.3: Performance on major false-belief benchmarks; accuracy (in %) unless otherwise stated. Parenthesis reflect differences between out-of-the-box model and fine-tuned version using ExploreToM-generated data. **Bold** reflects higher overall performance.

	ToMi	Hi-ToM	BigToM	OpenToM (F1)	FANToM
Llama-3.1 8B Instruct	68%	30%	75%	.39	0.3%
ExploreToM-8B	**95%** (+27)	**59%** (+29)	**81%** (+6)	**.46** (+.07)	0.2% (-0.01)

ries[1]. One key factor for this accuracy difference comes from models sometimes making the mental states more explicit through the infilling process: results shown correspond to a single attempt at infilling each story (41% of the samples ended successfully in a single attempt, judged by an LLM). Although stories remain challenging, since infilling with an LLM may introduce some ambiguities or hallucinations we only use them as training data. See App. D.3.2 for detailed results for all action sets.

5.4 ExploreToM is effective as training data generator

Experimental setup We fine-tune Llama-3.1 8B Instruct using a dataset of 79700 (story, question, answer) triples, focusing solely on the completion tasks, and dub the resulting model ExploreToM-8B. The dataset comprises both raw story structures and infilled stories, incorporating story structures from each of the 9 action sets listed in Table 5.1 (excluding asymmetry, and with a balanced number of questions per story type), and various user constraints—the same as in Section 5.3. We do full fine-tuning with the following hyperparameters: a learning rate of 10^{-6}, 100 warm-up steps, effective batch size of 40 samples, where we fine-tune solely on completions.

Fine-tuning with ExploreToM generalizes well to ExploreToM-generated data with more people and more actions than used in training Since ExploreToM-8B is trained with ExploreToM-generated data involving $p = \{2, 3, 4\}$ people with $m = \{2, 3, 4\}$ actions from the set of important actions $\mathcal{A}'$, we evaluate generalization within the ExploreToM domain by evaluating on ExploreToM-generated data involving 5 people, and up to 11 actions. This data is generated with Llama-3.1, the same model as original training data. See Figure 5.4.

Fine-tuning with ExploreToM improves or maintains performance on theory of mind benchmarks without hurting general reasoning capabilities We evaluate our fine-tuned ExploreToM-8B model on five prominent theory of mind benchmarks: ToMi (Le, Boureau, and Nickel, 2019), Hi-ToM (Wu et al., 2023), BigToM (Gandhi et al., 2024), OpenToM (Xu et al., 2024), and FANToM (H. Kim et al., 2023). Results show significant improvements in performance on ToMi and HiToM, with accuracy gains of +27

[1]Infilling can be also added to the A* search; we deemed it unnecessary given that this simpler method still yields a highly challenging benchmark and it is less costly.

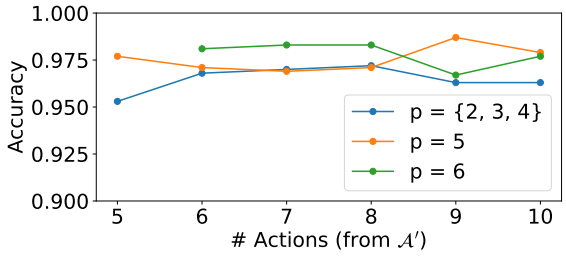


Figure 5.4: ExploreToM-8B accuracy when evaluating on ExploreToM-generated data with more people p and/or more actions a than seen during training ($p < 5, a < 5$). Performance remains high when adding several actions and/or up to two people.

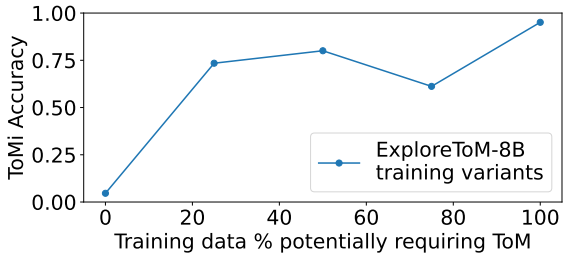


Figure 5.5: ToMi accuracy when training with ExploreToM-generated data with different proportions of interesting questions (i.e., questions potentially requiring theory of mind to answer). Here, all variants are fine-tuned with 85000 story structure samples for 1 epoch.

points on both benchmarks (see Table 5.3). The model maintains or shows small gains on the remaining three similar benchmarks, indicating that fine-tuning on ExploreToM data enhances or preserves performance across a range of theory of mind tasks[2].

We also evaluate out-of-domain reasoning skills using the two datasets: Multi3Woz (Hu et al., 2023), a commonly-used dataset for dialogue state tracking, and MMLU (Hendrycks et al., 2021), which tests both world knowledge and problem-solving abilities. Dialogue state tracking capabilities are preserved: both the base model and ExploreToM-8B achieve 96%. Broader reasoning capabilities are also generally preserved, with a small 2% performance difference (base model achieves 69%; ExploreToM-8B, 67%). Given the out-of-domain nature, we expect that intermixing data with samples more similar to MMLU's domains will substantially alleviate this slight regression.

Data mixture affects downstream performance We fine-tune five models, with 0%, 25%, 50%, 75%, or 100% of the stories requiring theory of mind to answer at least one question about the story. Figure 5.5 shows that training with as much stories that require theory of mind is crucial for achieving high downstream performance (using ToMi as a proxy dataset), even if some of the individual questions used for training do not require theory of mind.

5.5 On underlying skills needed for theory of mind

ExploreToM enables uncovering and quantifying underlying causes for models' poor theory of mind reasoning in models out-of-the-box. We specifically focus on the lack of robust state tracking skills, and the need for targeted training data in order to improve theory of mind capabilities.

[2]The lack of performance difference in FANToM is likely due to its length confounding factor: data points can have 1000+ tokens, yet our model is fine-tuned with a maximum length of 300 tokens.

LLMs lack robust state tracking skills ExploreToM's objective is to find story structures where models fail to answer questions; some of these questions simply require state tracking, specifically the ones where every person would give the same answer (i.e., their mental state is the same in this regard; e.g., in Fig. 5.1, all $X \in$ {Anne, Beth, Charles} would answer the same to "Where does X think Anne is right now?"). By definition (see § 5.2.2.2), these are the *uninteresting* questions. ExploreToM-generated questions are approximately evenly split between interesting and uninteresting, and uninteresting ones are even more challenging on average: the accuracy of interesting and uninteresting questions is 49% and 31% respectively for Llama-3.1 70B, 58% and 37% for GPT-4o, and 45% and 26% for Mixtral. See Table D.3 in App. D.3.3 for full breakdown for all settings.

State tracking questions are a subset of theory of mind questions, and arguably an easier case since the required logic for answering questions is simpler. Therefore, improving models' performance on state tracking may be a crucial prerequisite for achieving theory of mind reasoning in LLMs. As we have demonstrated, ExploreToM can be easily adapted to stress test pure state tracking, simply by retaining only the uninteresting questions.

Training data biases against theory of mind and its implications Figure 5.5 shows that to successfully improve performance on the ToMi benchmark, ExploreToM fine-tuning data needs to be biased towards *interesting* questions. However, a significant portion of models' training data is likely biased against requiring the tracking of divergent mental states (e.g., news articles).

As a conceptual proof that this phenomena occurs even within our custom domain-specific language unless we explicitly bias towards theory of mind, we demonstrate that randomly-sampled story structures tend not to require theory of mind. Using ExploreToM's domain-specific language, we randomly generate 1000 story structures with ToMi primitives ($\{a_{\text{enter}}$, a_{leave}, $a_{\text{moveObjContainer}}\}$) for stories involving $\{2, 3, 4\}$ people and $\{2, 3, 4\}$ object movements. We consider a story to not require theory of mind if all first-order and second-order theory of mind questions are *un-interesting*, as defined in § 5.2.2.2 (i.e., all share the same mental state). This stringent criterion evaluates *all* questions simultaneously. Nevertheless, our results show that 78% or more of the randomly-sampled stories meet this condition across all settings, with up to 87% of stories fulfilling the condition for the smallest setting (2 people, 2 object movements). When considering each question individually, 91%-95% are uninteresting questions. See App. D.3.4 for more details.

5.6 Related Work

Theory of mind benchmarking for language models Theory of mind benchmarks in language models can be categorized into human-generated and model-generated datasets. While human-generated datasets (Z. Chen et al., 2024; S. Kim et al., 2025; Shapira, Zwirn, and Goldberg, 2023) test reasoning about goals, emotions of others, and future actions, they are often limited in size and scope. Machine-generated datasets, such as foundational ToMi (Le, Boureau, and Nickel, 2019) and its successor Hi-ToM (Wu et al., 2023) focus primarily on mental state tracking, but have significant limitations: ToMi only supports a restricted set of actions ($\{a_{\text{enter}}$, a_{leave}, $a_{\text{moveObjContainer}}\}$), while Hi-ToM adds $a_{\text{info-}*}$ but

only as the last action in a story, and both datasets have extremely restricted interactions to orders. In contrast, our method, ExploreToM, significantly expands the scope of machine-generated datasets by supporting a larger number of actions, diverse wording, and plausible contexts. Unlike recent approaches that rely on LLMs for generation (Gandhi et al., 2024; H. Kim et al., 2023; Xu et al., 2024), ExploreToM ensures reliability and multi-interaction storytelling, making it a more comprehensive and robust benchmark for theory of mind in LLMs.

Theory of mind beyond language modeling Theory of mind has been explored beyond NLP in various settings, as reviewed in §4.6. Benchmarks developed since Chapter 4 have additionally evaluated theory of mind in multi-modal settings (Jin et al., 2024) and multi-agent collaboration (Bara, Sky, and Chai, 2021; H. Shi et al., 2025), but these focus on goal-driven interactions. Psychologists distinguish between affective (emotions, desires) and cognitive (beliefs, knowledge) theory of mind (Shamay-Tsoory et al., 2010), with cognitive theory of mind developing later in children (Wellman, 2014). Our work targets cognitive theory of mind, which is well-suited for generating situations with a domain-specific language and provides unambiguous answers across cultures. By focusing on cognitive theory of mind, our approach complements existing research and provides a comprehensive benchmark for this crucial aspect of human reasoning in language models.

Synthetic data generation Synthetic data has become a promising approach for acquiring high-quality data in various domains, including multihop question-answering (Lupidi et al., 2024) and language model evaluation (T. Wang et al., 2024). The process involves data augmentation/generation and curation, with techniques such as permutation-based augmentation (Q. Li et al., 2024; L. Yu et al., 2024) and iterative prompting (K. Yang et al., 2022). However, model hallucination (Guarnera, Giudice, and Battiato, 2020; Van Breugel, Qian, and Van Der Schaar, 2023; Wood et al., 2021; Y. Zhang et al., 2025) requires careful filtration and curation to ensure data quality. While prior works have used external feedback (Luo et al., 2024; Zelikman et al., 2022), our approach leverages an external LLM-as-judge to evaluate the plausibility and challenge of generated stories, both before and after infilling. Recently, AutoBencher (X. L. Li et al., 2025) has also been proposed to automatically search for datasets that meet a salience, novelty, and difficulty desiderata, highlighting the importance of careful benchmark creation. Unlike AutoBencher, which over-generates under the assumption that text-based conditioning minimizes hallucinations, our approach lifts this assumption and actively searches the space of possible narratives. This enables creating high-quality synthetic data regardless of the likelihood of a story being generated zero-shot, and generating even more challenging stories than with over-generation.

5.7 Conclusions

Theory of mind (ToM) is essential for social intelligence, and developing agents with theory of mind is a requisite for efficient interaction and collaboration with humans. Thus, it is important to build a path forward for imbuing agents with this type of reasoning, as well as methods for robustly assessing the of models' theory of mind reasoning capabilities.

We present ExploreToM, an A*-powered algorithm for generating reliable, diverse and challenging theory of mind data; specifically, creating synthetic stories that require theory of mind to understand them, along with questions to probe understanding. ExploreToM's adversarial nature enables the stress testing of future models and making our evaluation more robust to data leakage. We show that ExploreToM generates challenging theory of mind evaluation sets for many frontier models, with accuracies as low as 0% for Llama-3.1 70B Instruct and 9% for GPT-4o. Moreover, we show that ExploreToM can be used as a method for generating training data, leading to improvements of up to 29 accuracy points in well-known theory of mind benchmarks. Synthetic data is crucial for this domain, given that data that articulates theory of mind reasoning is difficult to find in the wild: children have access to a wide range of naturalistic social settings that incentivize the development of theory of mind but there is no such parallel pressure for LLMs.

Finally, we provide insights as to why basic theory of mind is still elusive to LLMs, including poor state tracking skills and demonstrating the need for training data that purposefully requires theory of mind, which is likely not present in the wild nor in randomly-generated data.

Limitations

ExploreToM offers a valuable tool for theory of mind research, and is a first step towards developing LLMs that can handle social interactions effectively. Although its data encompasses diverse and challenging settings—more than previously available—, and is grounded in established psychological tests, ExploreToM necessarily simplifies the complexity of real-world states and narratives by constraining it to the supported types of actions and interactions. Our framework requires manual coding of new actions, which can be a time-consuming process but comes with the benefit of a significant reliability improvement. Furthermore, our stories are not necessarily goal-oriented narratives, highlighting an important avenue for future work: creating datasets where actions stem directly from character goals to further enhance diversity and plausibility.

5.8 ExploreToM Remains Effective to Find Difficult Stories for 2026 Models

The original ExploreToM evaluation used strong models available in early 2025. Since then, frontier models have improved substantially. To assess whether ExploreToM retains its ability to surface challenging theory of mind stories, we execute it using Gemini 3.1 Flash Lite across all action sets and multiple combinations of number of people p and number of actions m.

Table 5.4 reports Gemini 3.1 Flash Lite results across all nine action sets and three difficulty levels, targeting $N = 30$ stories per setting with budget $B = 50$ per story. Flash Lite scores near zero on action sets involving room changes ($a_{\text{moveObjRoom}}$) and below .25 on most settings without communication actions, while action sets involving chitchat ($a_{\text{chitChat-*}}$) remain substantially easier—consistent with the original study.

Table 5.4: Accuracy of Gemini 3.1 Flash Lite (no thinking) on ExploreToM-generated stories, targeting $N=30$ stories per setting with budget $B=50$. Structure mirrors Table 5.1 but detailing problem sizes (p people, m movements).The $a_{\text{updateObjState}}$-only action set produced no valid stories under our constraints (—) and is included for completeness.

ExploreToM action set $\{a_{\text{enter}}, a_{\text{leave}}, \ldots$	$p=4$, $m=4$		$p=5$, $m=5$		$p=3$, $m=6$	
include asymmetry modifiers? (a_{peek}, $a_{\text{distracted}}$)	✗	✓	✗	✓	✗	✓
$\ldots, \boxed{a_{\text{moveObjContainer}}}\}$	.27	.15	.25	**.14**	.32	.20
$\ldots, \boxed{a_{\text{updateObjState}}}\}$	—	—	—	—	—	—
$\ldots, \boxed{a_{\text{moveObjContainer}}}, \boxed{a_{\text{updateObjState}}}\}$	.26	.10	.23	**.14**	.32	.15
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{moveObjRoom}}}\}$	**.03**	**.00**	**.00**	.23	**.00**	**.00**
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{info-*}}}\}$	.09	.13	.15	.22	.08	.03
$\ldots, a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}, \boxed{a_{\text{info-*}}}\}$	.23	.34	.42	.39	.19	.30
$\ldots, a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}, a_{\text{chitChat-*}}, \boxed{a_{\text{info-*}}}\}$	.78	.79	.81	.81	.82	.89
$\ldots, \boxed{a_{\text{chitChat-private}}}\}$	.91	.84	.94	.85	.92	.92
$\ldots, \boxed{a_{\text{chitChat-public}}}\}$	.57	.51	.58	.58	.78	.56

ExploreToM remains effective for reasoning models We selected six challenging settings for the non-reasoning model at $p=5$, $m=5$ and tested whether ExploreToM can find challenging stories for reasoning models. Table 5.5 shows that ExploreToM continues to surface genuinely difficult stories even for 2026 frontier models with extended reasoning such as Gemini 3.1 Pro.

Table 5.5: Accuracy on the six hardest action sets for Gemini 3.1 Flash Lite at $p=5$, $m=5$ ($N=10$ stories, $B=50$). Gemini 3.1 Flash Lite column is reproduced from Table 5.4 (top-10 out of 30) for direct comparison. Gemini 3 Flash and Gemini 3.1 Pro were run with chain-of-thought reasoning. Lower accuracy indicates ExploreToM successfully surfaced challenging stories for that model.

ExploreToM action set $\{a_{\text{enter}}, a_{\text{leave}}, \ldots$	Asym.	Gemini 3.1 Flash Lite	Gemini 3 Flash (reasoning)	Gemini 3.1 Pro (reasoning)
$\ldots, \boxed{a_{\text{moveObjContainer}}}\}$	✓	.00	.30	.68
$\ldots, \boxed{a_{\text{moveObjContainer}}}, \boxed{a_{\text{updateObjState}}}\}$	✓	.00	.44	.60
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{moveObjRoom}}}\}$	✗	.00	.45	.54
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{moveObjRoom}}}\}$	✓	.00	.55	.65
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{info-*}}}\}$	✗	.04	.41	.67
$\ldots, a_{\text{moveObjContainer}}, \boxed{a_{\text{info-*}}}\}$	✓	.12	.49	.69
Overall		.04	.43	.65

Part IV

Conclusions

5.9 Summary of Contributions

- I quantitatively established that LLM evaluation is highly sensitive to prompt formatting choices that preserve semantic meaning—a variance that persists regardless of model scale, few-shot examples, or instruction tuning, and that can reverse conclusions about which model performs better. I uncovered this through a structured approach that treats prompt formatting as a space defined by a grammar, enabling efficient search over semantically equivalent formats.

- I showed that transformer LLMs solve compositional tasks by reducing multi-step compositional reasoning into linearized subgraph matching, without necessarily developing systematic problem-solving skills. Their apparent success on compositional tasks is largely explained by exposure to training subgraphs that involve the same computations required for solving test instances, and their performance degrades as task complexity grows—a limitation that persists even when models are explicitly trained with step-by-step reasoning traces. I established this by recasting compositional tasks' algorithmic solutions as computation graphs, making complexity formally quantifiable and intermediate reasoning steps independently verifiable.

- I established theoretically that autoregressive error accumulation is provably exponential in compositional task complexity, and showed that the structure of a task predicts where models will partially succeed—explaining apparent competence as a consequence of leveraging shallow pattern-matching rather than the general reasoning ability the whole task would require.

- I showed that theory of mind reasoning in LLMs benchmark performance can be brittle to simple story structure perturbations, and that grounding inference in explicit symbolic belief-state representations enables robust inference-time improvements without any retraining. This work received the Outstanding Paper Award at ACL 2023. Building on this finding, I subsequently showed that frontier LLMs fail on theory of mind stories with identical reasoning requirements to examples they solve correctly, and established this through adversarial search over a structured space of narratives defined by a domain-specific language for mental state tracking, removing the need to manually construct perturbations and making the evaluation robust to data leakage in a way that static benchmarks cannot be.

- I demonstrated that structured data generation can simultaneously achieve formal verifiability and linguistic diversity—properties typically in tension—for theory of mind reasoning, and that fine-tuning on such data yields substantial improvements on established benchmarks while preserving general reasoning capabilities. I achieved this by grounding story generation in a domain-specific language for mental state tracking, enabling A* search to efficiently navigate the space of possible narratives, while leveraging LLMs to sample diverse surface realizations of the same underlying mental state updates.

5.10 Conclusions

As language models are deployed in increasingly consequential settings, the ability to robustly characterize their capabilities and failure modes becomes critical. Without explicit structure to anchor our analyses, evaluation may conflate prompt artifacts with genuine capability, benchmark success may mask systematic failures, and improvements may prove fragile and uninterpretable. We demonstrated, across multiple reasoning domains, that reintroducing structure at both the surface and semantic levels restores the analytical leverage needed for principled evaluation, higher-quality data generation, and interpretable inference-time improvement.

Regarding surface-level structure, **FormatSpread** established that LLM evaluation is confounded by a largely invisible source of variance: the precise typographic and syntactic choices used to format a prompt. By treating the space of semantically equivalent prompt formats as a grammar, we enabled efficient multi-armed bandit search over this space, revealing performance swings large enough to reverse conclusions about model rankings, with no change in the underlying semantic content of the prompt.

Regarding semantic-level structure, **Faith and Fate** recasted compositional reasoning tasks as computation graphs, making reasoning steps formally quantifiable and independently verifiable. The resulting analyses revealed that apparent success on these tasks is largely explained by pattern-matching against subgraphs seen during training, and that performance decays with task complexity—a trend we further grounded theoretically by proving that autoregressive error accumulation is exponential in compositional task complexity.

In the domain of social reasoning, **SymbolicToM** showed that theory of mind benchmark performance can be severely brittle to story structure perturbations, and that grounding inference in explicit symbolic belief-state representations yields dramatic and more robust improvements without any retraining. **ExploreToM** then addressed data generation: by defining a domain-specific language for mental state updates and applying A* search over the resulting narrative space, it produced theory of mind scenarios that are simultaneously formally verifiable and linguistically diverse (thanks to its LLM-augmented approach), with fine-tuning on this data yielding substantial gains on established benchmarks. Together, these works illustrate how semantic-level structure can advance all three impact areas: it exposed benchmark limitations, produced higher-quality training data, and improved inference-time reasoning, all within the same formal apparatus.

Across these works, a consistent pattern emerges. Structure, whether imposed over the typographic surface of a prompt or over the semantic content of a reasoning task, serves as a multiplier: it enables search algorithms that would be inapplicable to unstructured data, makes failure modes verifiable rather than merely observable, and creates a virtuous cycle in which better evaluation leads to stronger modeling. The theory of mind works make this cycle most explicit, with structured evaluation exposing benchmark limitations, structured generation producing training data that addresses those limitations, and structured inference improving model behavior without retraining.

Exploring local search algorithms across both works reveals an instructive contrast: FormatSpread found the surface of prompt format space so jagged that local search was ineffective (Chapter 2), while ExploreToM found that local search over semantically meaningful perturbations substantially outperformed the more standard approach of overgenerating

and filtering. This contrast suggests that the nature of the perturbation matters: when modifications induce significant and more predictable changes in internal representations, the resulting space is amenable to search in a way that purely surface-level perturbations currently are not.

It is worth contextualizing the studies presented here to properly understand their intention and research contribution, particularly in a landscape of rapidly improving models. The tasks studied in **Faith and Fate** were chosen to be simple task instantiations that cleanly isolated compositional behavior and minimized confounders. It is thus expected for models to eventually solve the concrete task instances analyzed as scale increases; the goal was never to define an insurmountable frontier, but to study the nature and rate of performance decay as a function of task complexity. Shojaee et al. (2025) have since found analogous patterns in large reasoning models, which exhibit accuracy collapse beyond certain complexity thresholds and a counterintuitive pattern in which reasoning effort ceases to scale with problem complexity despite available token budget. Taken together, these results suggest that the relationship between apparent reasoning competence and generalizable problem-solving ability remains an open question even in the most recent models. At the same time, it is worth noting that for many practical applications, solving tasks only up to moderate instance sizes may be entirely sufficient, and that memorization can itself be a valuable capability when the distribution of instances encountered in deployment is narrow enough.

On the formatting side, posterior research building directly on FormatSpread continues to uncover the same behaviors in models released well after the original work. Su et al. (2025) showed that a single delimiter character suffices to substantially alter model rankings; Feng et al. (2025) demonstrated analogous brittleness in vision-language models, where superficial visual choices alone suffice to reverse leaderboard rankings; and Romanou et al. (2026) found ranking instability in the majority of pairwise model comparisons under prompt perturbation. Format sensitivity appears to be a systemic property of current models rather than an artifact of any particular model generation.

5.11 Future Work

Automatically imbuing tasks with semantic-level structure. A central motivation for the structured approaches developed in this thesis is the brittleness of static evaluation datasets: once a dataset leaks into pretraining, the effort invested in constructing it is effectively discarded. Algorithmically generated evaluation data offers a principled response, since a generation procedure cannot be memorized as easily as specific instances can. Moreover, the accelerating pace of capability development motivates a broader shift toward benchmark generation procedures whose difficulty can be scaled programmatically, so that evaluation keeps pace with the models being evaluated. As observed in the conclusions, the behaviors documented in this thesis persist in recent models but tend to surface at larger problem sizes—making it essential that benchmarks can be recalibrated rather than discarded as the frontier moves.

One promising direction is to have an LLM agent automatically write code that generates evaluation data in a task-agnostic way, minimizing the manual effort required to specify semantic-level structure. Such an approach is already viable in verifiable domains such as

code and mathematics, where program outputs can be checked automatically, and holds considerable potential for natural language settings where direct execution is not available. However, as automated evaluation procedures are increasingly repurposed as reinforcement learning environments, a new threat emerges: a model may learn to reward-hack the evaluation metric without acquiring the underlying capability it was designed to measure. The key design requirement is therefore that the generated programs encode correctness in a form that cannot be gamed through surface-level optimization, making the generation pipeline itself the primary source of quality assurance, and making the generated programs sampled diverse enough to cover the diversity of the underlying real task.

Toward evaluation beyond verifiable tasks. Verifiable tasks are the most natural fit for algorithmically scalable evaluation, as their correctness criteria are directly compatible with reinforcement learning-based difficulty scaling. Social reasoning, and theory of mind in particular, remains the harder case: the space of valid responses is often genuinely broad, and not every correct inference corresponds to a single checkable answer. During this thesis, structured domain-specific languages for mental state tracking made it possible to define subsets of theory of mind tasks with unambiguous answers, and there remains room to extend benchmark generation within this paradigm.

More broadly, when the synthetic data generation pipeline itself encodes the answer—as in FANToM (H. Kim et al., 2023) and FlawedFictions (Ahuja, Sclar, and Tsvetkov, 2025), two of my collaborations—verification becomes tractable even in social reasoning domains. Future work will ultimately require to venture into scenarios where multiple responses are valid, developing verification procedures that can assess correctness without requiring a unique ground truth, and that can handle ambiguity.

Building models that are robust and interpretable by design. The works in this thesis are primarily diagnostic and generative: they surface failures, explain them structurally, and address them through inference-time algorithms and targeted fine-tuning. A complementary direction is to address brittleness from the modeling side, by training models whose internal representations are more robust across semantically equivalent inputs, whether through augmenting training data with systematically varied prompt formats or through objectives that explicitly encourage invariance to surface-level perturbations. More broadly, the dominant paradigm of scaling without structural inductive biases has yielded remarkable capabilities, but the brittleness documented here suggests that some degree of structural constraint should help. Even flexible decompositions can yield meaningful gains: S. S. Li et al. (2025) showed that decomposing preferences into interpretable attribute dimensions, and modeling how different communities weight those attributes, produces both stronger preference prediction and more transparent insights than aggregate modeling alone, illustrating that structure and scale can be complementary rather than competing paradigms.

Discovering interpretable predictors of prompt quality. As observed in the conclusions, FormatSpread found the surface of spurious prompt format space too jagged for principled navigation, yielding no concrete recommendations for what makes a format reliably good. An important direction for future work is to move away from the pure dichotomy

between prompts that carry exactly the same meaning and prompts that differ semantically, and instead characterize the space of predictable, measurable changes one can make to a prompt and study their impact on model behavior. The contrast with ExploreToM is instructive: local search over semantically meaningful perturbations proved substantially more effective than for surface-level ones, suggesting that more meaningful features—those that induce predictable changes in internal representations—may define smoother spaces that are genuinely amenable to structured search. This raises the prospect of identifying interpretable features of prompt design, such as how information is ordered, how context is decomposed, or how constraints are expressed, that reliably predict model performance. Such features could serve both as practical tools for prompt optimization and as a basis for more reliable evaluation upper bounds, ensuring that capability assessments measure what models can genuinely do rather than what their prompts happen to elicit.

Part V

Appendices

Appendix A

FormatSpread: Grammar Details and Additional Experiments

A.1 Grammar Definition and Instantiation Details

A.1.1 Equivalence Relation Definition

Precisely, $p_1 \sim p_2$ if and only if at least one of the following hold: $p_1 = p_2 = B_0$; or $p_i = B_0'(d_i, s_i)$ with $d_1 = d_2$; or $p_i = B_1(d_i, s_i, f_i)$ with $d_1 = d_2$; or $p_i = B_2^{(n)}(X_{1,i}, \ldots, X_{n,i}, c_i)$ with $X_{j,1} \sim X_{j,2}$ $\forall 1 \leq j \leq n$; or $p_i = B_3^{(n)}(d_i, j_{1,i}, \ldots, j_{n,i}, s_1, s_2, c, f)$ where $d_1 = d_2$ and $j_{k,1} = j_{k,2}$ $\forall 1 \leq k \leq n$. It is possible that generated formats equivalent in their string representation are not equivalent according to this equivalence relation.

A.1.1.1 Visualization of Prompt Format's Parsing and Full Format Generation

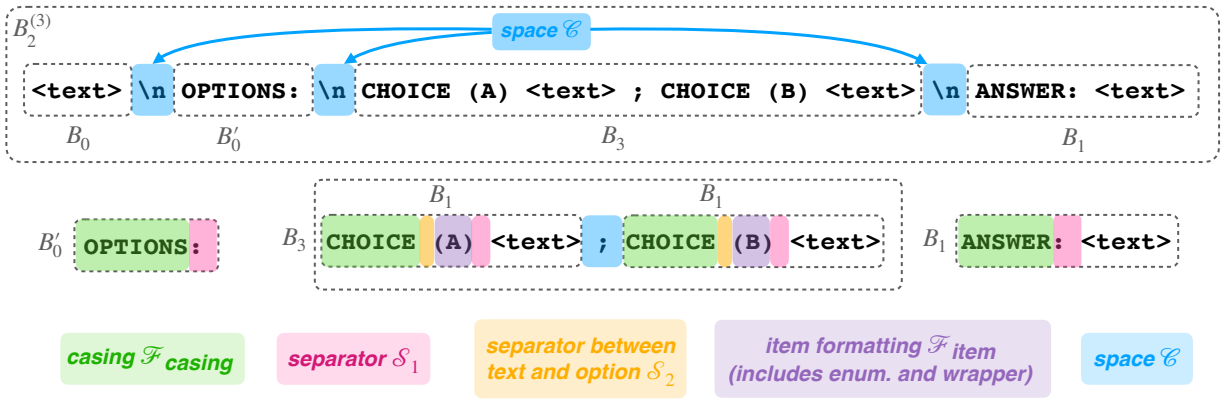


Figure A.1: Visualization of a complex prompt format showing its parsing and which constants or functions affect each part of the format.

Figure A.1 shows a visualization of how a complex format is parsed using our defined grammar. A full prompt consists of an instruction, n few-shots and a data point to solve.

For example, if the instruction was `Given a sentence and two words that appear in it, answer which one of the two (A or B) appeared first in the sentence.`, a full prompt may look as follow. Note that we always use `\n\n` as space character between instruction and few-shots. The example below shows a 1-shot prompt. It is simply illustrative and does not correspond to any of the tasks considered.

```
Given a sentence and two words that appear in it, answer which one of the two
(A or B) appeared first in the sentence.

The quick brown fox jumps
OPTIONS:
CHOICE (A): fox ; CHOICE (B): brown
ANSWER: B

Over the lazy dog
OPTIONS:
CHOICE (A): lazy ; CHOICE (B): dog
ANSWER:
```

FormatSpread forces all instantiations of a multiple choice variable to change jointly to maintain coherence, and this includes text in the instruction. Therefore, when changing the option items from A and B to I and II, the prompt will be generated as follows.

```
Given a sentence and two words that appear in it, answer which one of the two
(I or II) appeared first in the sentence.

The quick brown fox jumps
OPTIONS:
CHOICE (I): fox ; CHOICE (II): brown
ANSWER: II

Over the lazy dog
OPTIONS:
CHOICE (I): lazy ; CHOICE (II): dog
ANSWER:
```

A.1.2 Allowed values for each set $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{C}$, $\mathcal{F}_{\text{casing}}$, $\mathcal{F}_{\text{item}}$

$$\mathcal{S}_1 = \{'','\ ','\backslash n','\backslash n','--','\ ','; \backslash n','||','<\texttt{sep}>','--','\ ','\backslash n','\ ',',','\backslash n','.','\ ','\ '\}$$
$$\mathcal{S}_2 = \{'','\ ','\ \ ','\backslash t'\} \text{ (no space, single space, double space, tab)}$$
$$\mathcal{C} = \{'',':::','::','\ :','\ :\ ','-','\backslash n\backslash t','\backslash n\ \ ','\backslash n','\backslash n\backslash t',':','::','-','\backslash t'\}$$
$$\mathcal{F}_{\text{casing}} = \{\texttt{f(x)} = \texttt{x}, \texttt{f(x)} = \texttt{x.title()}, \texttt{f(x)} = \texttt{x.upper()}, \texttt{f(x)} = \texttt{x.lower()}\}$$
$$\mathcal{F}_{\text{item}} = \{x \mapsto f(g(x)) \mid \text{ such that } f \in \mathcal{F}_{\text{item1}} \wedge g \in \mathcal{F}_{\text{item2}}\}$$
$$\mathcal{F}_{\text{item1}} = \{\texttt{x} \mapsto (\texttt{x}), \texttt{x} \mapsto \texttt{x.}, \texttt{x} \mapsto \texttt{x}), \texttt{x} \mapsto \texttt{x}), \texttt{x} \mapsto [\texttt{x}], \texttt{x} \mapsto <\texttt{x}>\}$$
$$\mathcal{F}_{\text{item2}} = \{\texttt{x} \to \texttt{x} + 1, \texttt{x} \to' \texttt{A}' + \texttt{x}, \texttt{x} \to' \texttt{a}' + \texttt{x},$$
$$\texttt{x} \to \texttt{0x215F} + \texttt{x} + 1, \texttt{x} \to \texttt{ROMAN[x].lower()}, \texttt{x} \to \texttt{ROMAN[x].upper()}\}$$

Enumerations are indexed from (i.e., "1, 2, 3" rather than "0, 1, 2"). `ROMAN[x]` represents the Roman numerals written in regular ASCII characters. `'0x215F'+x` represent the series of Unicode characters for Roman numerals. ␣ denotes a spacing character for clarity.

A.1.3 Restrictions to Prompt Formats Spaces and Separators' Combinations

We define several restrictions to ensure format naturalness. Users can additionally customize FormatSpread by defining their own rules and restrictions between values. Our rules are as follows:

- If $B_2(X_1,\ldots,X_n,c)$ where c does not contain a newline, then each X_i's separators and any subcomponents' separators should not contain a newline.
- Similar to the rule above, if $B_3^{(n)}(d, j_1, \ldots, j_n, s_1, s_2, c, f_1, f_2)$ such that some separator contains a newline (i.e. s_1 contains a newline and/or s_2 contains a newline) then the space c must also contain a newline.
- For $B_1(d, s, f) := f(d)s<$ `text` $>$, s must not be the empty string (i.e., there has to be some separation between descriptor and text).
- Having c be an empty string space in $B_2^{(n)}$ is only allowed if the first $n-1$ components are B_1 fields with an empty `<text>`. Similarly, the newline restrictions mentioned above only apply if the `<text>` is not empty. This rarely happens in prompt formats, but there are formats such as `Question: <text> Options: A. <text> B. <text>` where the `Options:` do not have a corresponding field.

A.1.4 Thompson Sampling Priors

For the first exploration (i.e., finding the best-performing prompt format), we set an informative prior $\text{Beta}(\alpha, \beta) := \text{Beta}\left(\max\left(\frac{\beta \cdot x}{1-x}, 1.1\right), 5\right)$ for all arms p_i, where x is the original format's accuracy. Our goal is to set an informative prior where the expected value of the prior distribution is the original format accuracy x, since a priori it is the only information we have about performance. This restricts the parameters as follows:

$$\begin{aligned}\mathbb{E}[\text{Beta}(\alpha, \beta)] = \frac{\alpha}{\alpha + \beta} &= x \\ \alpha &= \alpha \cdot x + \beta \cdot x \\ \alpha &= \frac{\beta \cdot x}{1 - x}\end{aligned}$$

Since β will modulate how confident is the prior, and we want to avoid the model being overconfident, we fix $\beta = 5$. Because we want to have an informative prior $\text{Beta}(\alpha, \beta)$ with a Gaussian-like PDF, we force $\alpha > 1$ and $\beta > 1$. In extreme cases, forcing $\alpha > 1$ might alter the expected value. The first exploration's priors are thus exactly $\text{Beta}(\alpha, \beta)$ with $\alpha = \max\left(\frac{\beta \cdot x}{1-x}, 1.1\right)$ and $\beta = 5$ for all arms p_i.

For the second exploration (i.e., finding the worst-performing prompt format), the model has access to the first explorations' counters $S_i^{(E/B)}$ and $N_i^{(E/B)}$. Therefore, we set the second exploration's priors to be Beta $\left(\alpha + S_i^{(E/B)}, \beta + \left(N_i^{(E/B)} - S_i^{(E/B)}\right)\right)$.

A.2 Additional Experiments' Information and Plots

A.2.1 Task selection

We use a number of heuristics to filter Super-NaturalInstructions tasks to our set of 53 evaluation tasks. Datasets should have at least 1000 samples to be considered. We also remove tasks whose instructions are too long (over 3,000 characters) and datasets with inputs longer than 2,000 characters, given that this makes performing inference at scale intractable. We also filter datasets whose valid outputs include more than 20 different strings, given that we focus on classification tasks.

We also removed tasks where we found a priori performance on the task was 0% accuracy using LLaMA-2-7B 1-shot. Some Super-NaturalInstructions tasks are derived from the same original dataset, but ask different questions. We did not include more than 4 tasks from the same original dataset.

Finally, we also searched for having socially impactful tasks. Those tasks were the only Super-NaturalInstructions tasks where we included a format if one was not provided by the dataset.

The selected tasks were the following 53: `task050`, `task065`, `task069`, `task070`, `task114`, `task133`, `task155`, `task158`, `task161`, `task162`, `task163`, `task190`, `task213`, `task214`, `task220`, `task279`, `task280`, `task286`, `task296`, `task297`, `task316`, `task317`, `task319`, `task320`, `task322`, `task323`, `task325`, `task326`, `task327`, `task328`, `task335`, `task337`, `task385`, `task580`, `task607`, `task608`, `task609`, `task904`, `task905`, `task1186`, `task1283`, `task1284`, `task1297`, `task1347`, `task1387`, `task1419`, `task1420`, `task1421`, `task1423`, `task1502`, `task1612`, `task1678`, `task1724`.

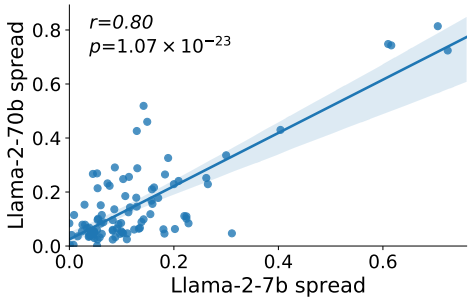


Figure A.2: Comparison between Llama-2-7B and Llama-2-70B spreads. Llama-2-70B was computed using 4bit quantization (Dettmers et al., 2022).

A.2.2 Additional Results for Section 2.4.2

Table A.1: Ratio of prompt format pairs (p_1, p_2) such that if p_1 is worse than p_2 using model M_1, then the same trend holds for M_2.

Model 1 (M_1)	Model 2 (M_2)	Performance Relative Ordering Preservation
Llama-2-7b	Llama-2-13b	57.46%
Llama-2-7b	Falcon-2-7b	55.91%
Falcon-7b	Falcon-7b-Inst	61.11%

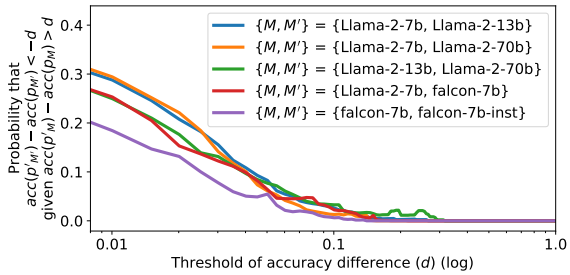


Figure A.3: Option ranking metric.

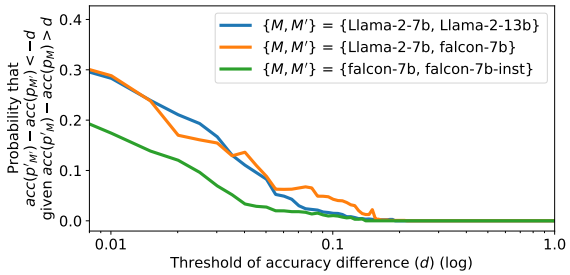


Figure A.4: Exact prefix matching metric.

Figure A.5: Probability of a prompt p being worse than p' by at least d points using model M', given that prompt p was *better* than prompt p' when using model M.

Formats are not inherently good or bad. Table A.1 shows that if format p_1 has lower performance than format p_2 under model M, there is < 0.62 probability that this trend would hold under another model M' (random chance is 0.5). This weak relative order preservation suggests that prompt format performance in a model may not be extrapolated to a different model, or in other words, that there are no inherently good or bad formats. This finding is further supported by Figure A.5, which shows that findings of a format being better or worse than another are often inconsistent across models.

Experiments with exact prefix matching accuracy. Here we show results with using exact prefix matching to compute accuracy. Often, failures in prefix matching are associated with degeneration, i.e., cases where the model does not answer any of the valid options, motivating the use of ranking accuracy. Degeneration makes models (specially smaller models) more unlikely to have high accuracy out of the box. As seen in Figure 2.9, prefix matching is linked to having higher changes when performing atomic changes. Moreover, exact prefix matching can lead to lower performance as generation is less constrained (see Figure A.14). Table A.2 shows examples of atomic changes yielding large accuracy changes with exact prefix matching metric.

Figure A.8 shows spread remains regardless of model size increase, architecture change, or number of few-shot examples also when using exact prefix matching as accuracy metric. In line with the results shown for probability ranking in Section 2.4.2, Figure A.11 shows that the probability of reversing performance trends between two models just by changing prompt remains high when using exact prefix matching as metric. Strikingly, spread is significantly higher than in the probability ranking setting (see Figure A.10), with median spread ranging from 12 to 28 accuracy points depending on the model used. This further motivates the need for running FormatSpread when benchmarking models with this accuracy metric. This increased spread may be partly due to degeneration, as we will detail next.

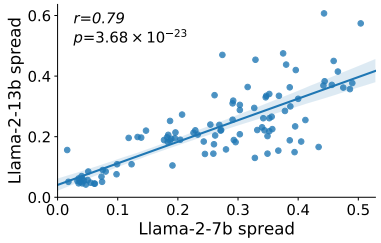


Figure A.6: Llama-2-7B vs. 13B

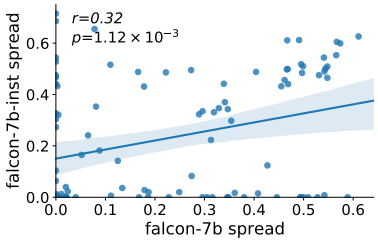


Figure A.7: Falcon-7B vs. 7B-Instruct

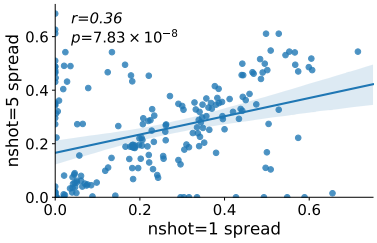


Figure A.8: 1- vs. 5-shot (same task, model)

Figure A.9: Spread comparison between evaluating the same task under different models or n-shots using exact prefix matching as accuracy metric.

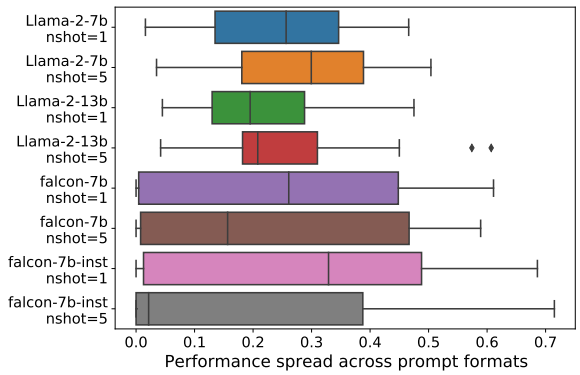


Figure A.10: Spread across models and n-shots. Exact prefix matching metric.

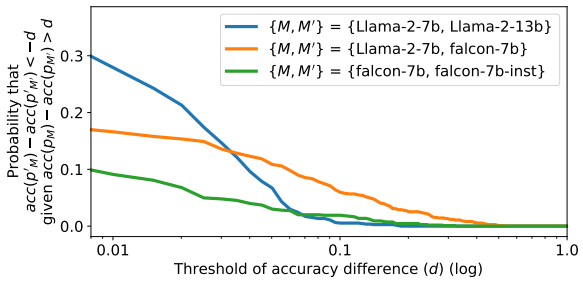


Figure A.11: Probability that model M performs *worse* than M' by at least d when using format p', given that M performed *better* than M' by at least d using format p. 53 tasks, 1- and 5-shot. Exact prefix matching metric.

Degeneration. Sometimes when a model does not generate the correct answer with exact prefix matching, it also does not generate a valid response, i.e. it degenerates. We will now quantify this phenomenon using 53 SuperNaturalInstructions classification and multiple choice tasks.

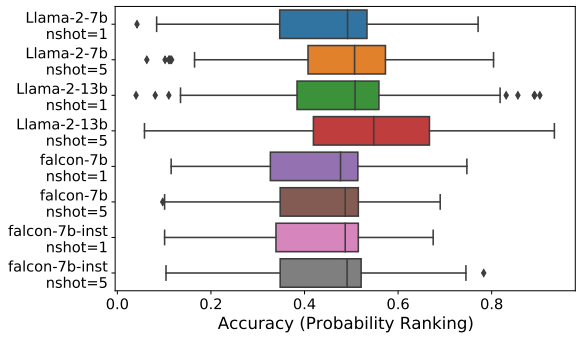


Figure A.12: Accuracy boxplot for the selected 53 Super Natural-Instructions tasks, option ranking metric.

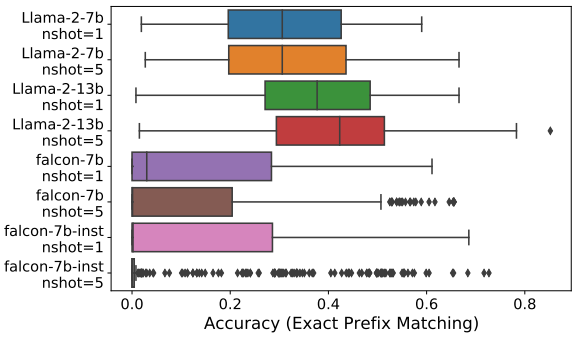


Figure A.13: Accuracy boxplot selected 53 Super Natural-Instructions tasks, exact prefix matching metric.

Figure A.14: Accuracy metric used can strongly impact final performance. 53 Super Natural-Instructions tasks shown. Ranking accuracy yields higher accuracies overall.

Given a model, a task, and a format, let the *centered mass* be the ratio of examples where the model's output matched with any valid option (regardless of correctness). Table A.3 shows that the correlation between accuracy and centered mass is moderate or high depending on the model. This suggests that very often when a model does not return a valid answer, it does not return any valid answer at all. This is especially true for Falcon models, where we observe an almost perfect correlation between accuracy and centered mass. In conclusion, prompt format chosen often do not solely affect accuracy, but they also affect the frequency in which a model is actually able to perform a task. This will especially affect tasks for which there are no alternative metrics. Further research may focus specifically on targeting features that cause degeneration.

Experiments with Instruction Induction tasks. All experiments thus far focused solely on classification tasks. We will now focus on tasks that require generating (short) text, and cannot be framed as classification tasks. We selected 10 tasks from Instruction Induction (Honovich et al., 2023) that require generating a unique, valid string to be considered a correct response. Examples include identifying the second letter of a word, adding numbers, or answering a synonym to a given word. Instruction Induction tasks also show a wide range of difficulty, resulting in varied settings to be analyzed (see Figure A.20). Given that the collection does not contain human-generated formats, we applied a simple '`Input: {}\n Output: {}`' format. Results for 1-shot and 5-shot settings show spread is still high across models and n-shot choices (see Figure A.18).

Tasks are: `antonyms`, `diff`, `first_word_letter`, `larger_animal`, `letters_list`, `num_to_verbal`, `second_word_letter`, `singular_to_plural`, `sum`, `synonyms`.

Experiments with continuous metrics in open-ended text generation tasks. Throughout the paper we focus on tasks with a single valid output, whether in classification tasks or in short-text generation tasks. This decision is intentional, since it guarantees

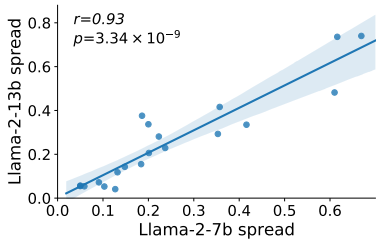


Figure A.15: Llama-2-7B vs. 13B

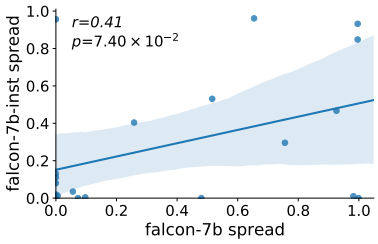


Figure A.16: Falcon-7B vs. 7B-Instruct

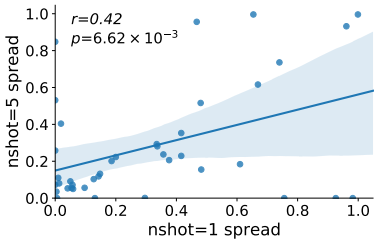


Figure A.17: 1- vs. 5-shot (same task, model)

Figure A.18: Spread comparison between evaluating the same task under different models or n-shots for Instruction Induction tasks. Exact prefix matching used as accuracy metric.

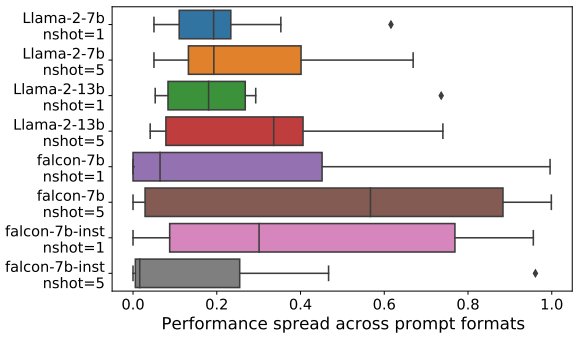


Figure A.19: Spreads across models and n-shots.

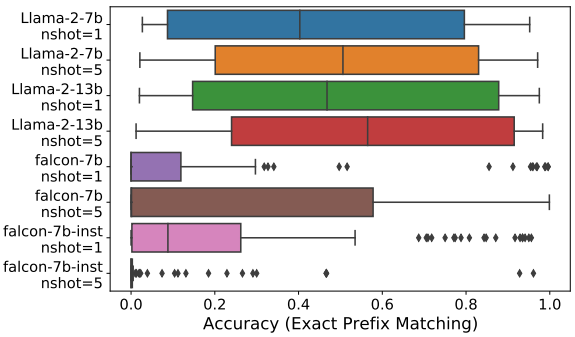


Figure A.20: Accuracy variance by model

Figure A.21: Instruction Induction tasks' spreads and accuracy across models. Exact prefix matching is used as accuracy metric.

Table A.2: Examples of atomic changes' impact on accuracy using prefix matching (probability ranking shown in Table 2.2). `{}` represents a text field; p_2 yields higher accuracy than p_1 for all tasks.

Task Id	Prompt Format 1 (p_1)	Prompt Format 2 (p_2)	Acc p_1	Acc p_2	Diff.
task213	`Title: {} Sentence<A>: {} || Sentence<B>: {} || Sentence<C>: {} || Sentence<D>: {} Choices: \n <i>::: {} \n <ii>::: {} Answer: {}'`	`Title::{} Sentence<A>::{} || Sentence<B>::{} || Sentence<C>::{} || Sentence<D>::{} Choices::\n <i>::: {} \n <ii>::: {} Answer::{}'`	0.113	0.475	0.362
task296	`Sentence I ) : {} \nSentence II ) : {} \nSentence III ) : {} \nSentence IV ) : {} \nSentence V ) : {} \nSentence VI ) : {} \nSentence VII ) : {} \nSentence VIII ) : {} \nSentence IX ) : {} \nSentence X ) : {} , I. : {} , II. : {} , Answer: {}'`	`Sentence I ) : {} \nSentence II ) : {} \nSentence III ) : {} \nSentence IV ) : {} \nSentence V ) : {} \nSentence VI ) : {} \nSentence VII ) : {} \nSentence VIII ) : {} \nSentence IX ) : {} \nSentence X ) : {} , 1. : {} , 2. : {} , Answer: {}'`	0.201	0.522	0.321
task905	`Tweet::: {}; \nLabel::: {}; \nAnswer::: {}'`	`Tweet::{}; \nLabel::{}; \nAnswer::{}'`	0.252	0.559	0.307
task317	`Passage:: {} \nAnswer:: {}'`	`Passage::{} \nAnswer::{}'`	0.245	0.546	0.301
task280	`passage {}\n answer {}'`	`passage:{}\n answer:{}'`	0.332	0.612	0.28
task050	`SENTENCE - {} \nQUESTION - {} \nANSWER - {}'`	`SENTENCE\n\t{} \nQUESTION\n\t{} \nANSWER\n\t{}'`	0.244	0.504	0.26
task070	`Beginning - {}\nMiddle [I]{} , Middle [II]{}\nEnding - {}\nAnswer - {}'`	`Beginning - {}\nMiddle I){} , Middle II){}\nEnding - {}\nAnswer - {}'`	0.143	0.3	0.157

that a variation in the metric truly represents a variation in model performance. We have shown that spread remains high when considering option ranking or exact prefix matching as accuracy metric.

Since LLMs are often used in more open-ended generation contexts, we will now explore the performance variance across prompt formats when considering sentence-length generation tasks (e.g. generate the next sentence of a story, given the four initial sentences of a story, generate a question whose answer is the sentence given). To analyze the automatic generations, we use two widely used metrics: ROUGE-L (C.-Y. Lin, 2004), and BERTScore (T. Zhang et al., 2019). The first is an n-gram-based metric, and the latter is a model-based metric, and both are $[0, 1]$ metrics where higher is better. Figure A.22 shows that variance remains high for LLaMA-2-7B regardless of the metric and the number of n-shots considered, with LLaMA-2-7B 5-shot having 25% of tasks with a ROUGE-L spread of 0.098 or higher, and a BERTScore spread of 0.09 or higher.

We observe that the median spread is sometimes smaller than in the accuracy tasks. This may be because although ROUGE, BERTScore, and accuracy are all $[0, 1]$ metrics, typical metric values may be different, which may in turn affect the final spread (an absolute difference). We leave it to future work to quantify the differences in style or content that each format may be inducing.

Finally, it is worth noting that text generation metrics are known to be noisier, and thus not all metric decreases necessarily correspond to a true performance loss, as is the case for accuracy in single-valid-output tasks. We used 17 SuperNatural-Instructions tasks: `task037`, `task038`, `task040`, `task067`, `task071`, `task072`, `task105`, `task216`, `task223`, `task240`, `task348`, `task389`, `task443`, `task845`, `task1326`, `task1401`, `task1613`. We selected the 17 open-ended text generation tasks among those with at least 1000 samples, with some formatting present in the original task (e.g. `'Passage:' <text>`). We only considered tasks whose instructions were under 1,000 characters and that

Table A.3: Correlation between accuracy using exact prefix matching and the centered mass (the opposite of degeneration). 53 tasks, 10 formats each, evaluated on 1000 samples.

Model	n-shot	correlation between accuracy & and centered mass	p-value
Llama-2-7b	1	0.702	5.1E-77
Llama-2-7b	5	0.762	4.9E-98
Llama-2-13b	1	0.639	5.8E-61
Llama-2-13b	5	0.662	9.2E-67
falcon-7b	1	0.936	7.1E-233
falcon-7b	5	0.933	8.4E-228
falcon-7b-instruct	1	0.962	3.6E-289
falcon-7b-instruct	5	0.958	5.5E-277

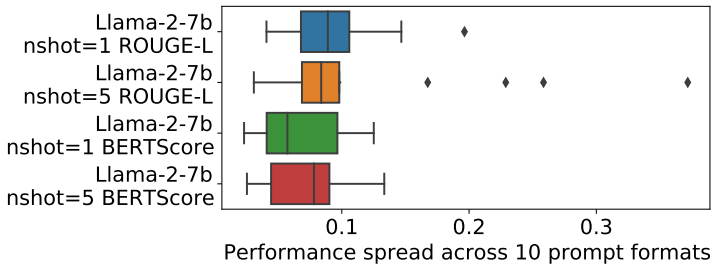


Figure A.22: Spread across n-shots for LLaMA-2-7B, considering ROUGE-L and BERTScore metrics. 17 sentence-level open-generation tasks are considered, all extracted from SuperNatural Instructions. 10 prompt formats are considered for each task.

contained inputs no longer than 5,000 characters.

We limit generations to 50 tokens. To parse model outputs more faithfully, and given that none of our expected generations include a newline, we only consider a model's generation up to the first newline (excluding leading spaces and newlines in a given generation). This consideration is important given that often models start to generate a new data sample from scratch, immediately after generating the requested answer.

Characterizing a model's accuracy distribution beyond spread. Spread gives a quantitative jump in information with respect to informing a single point in the performance distribution since it measures the distribution range (maximum minus minimum). However, distributions that may share the same range, may yield a widely different probability of obtaining each value in the distribution. Figure A.23 plots the accuracy distribution of 30 tasks, sorted in decreasing order by standard deviation. Tasks with high standard deviation reflect a higher likelihood of obtaining dissimilar values when making a formatting selection; Figure A.23 shows that the median standard distribution is $\sigma \approx 0.04$, which can be considered high in our context.

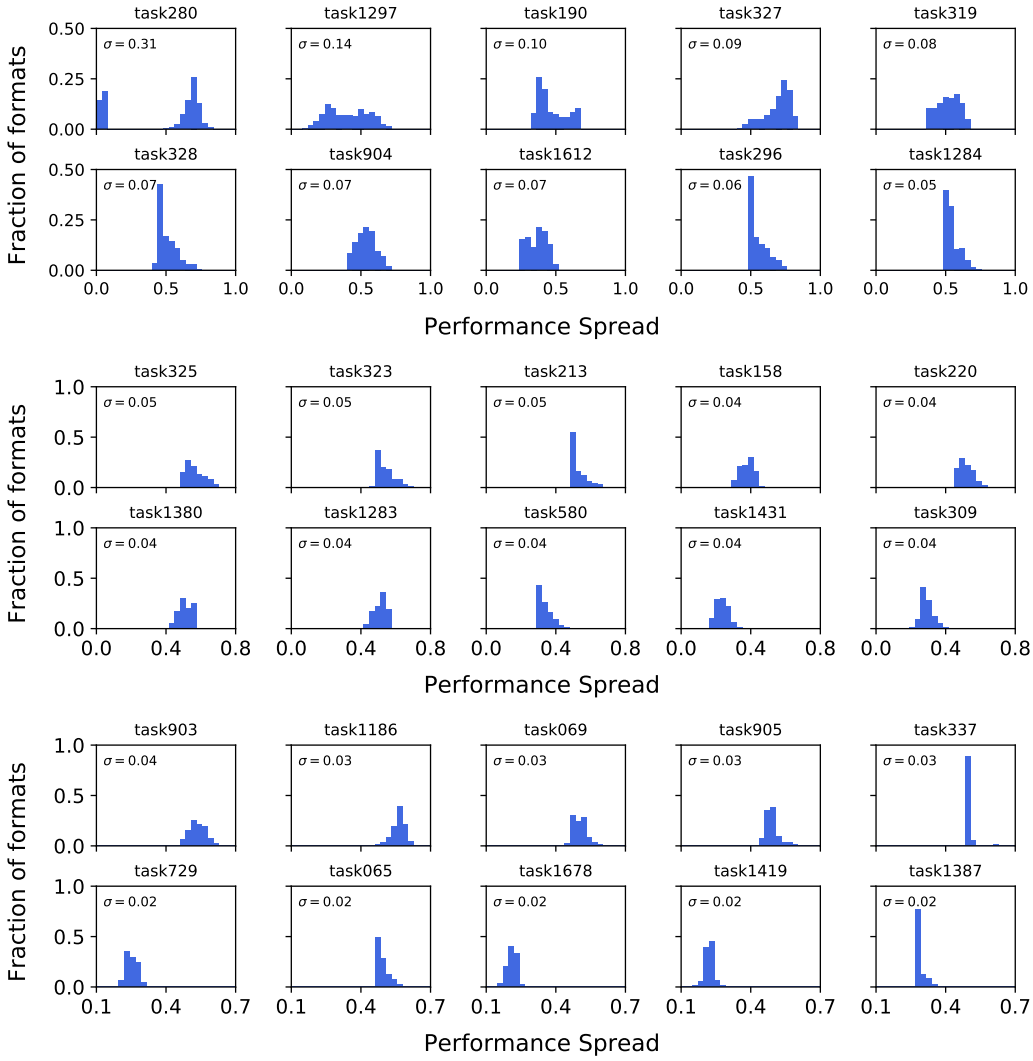


Figure A.23: Accuracy distribution across 500 formats for 30 tasks evaluated on 250 samples each, sorted by standard deviation in decreasing order. LLaMA-2-7B 1-shot, option ranking metric.

On factors influencing spread besides prompt formatting. We believe many factors beyond formatting may be influencing performance variance, but were unable to find a feature that reliably predicts spread. We found that the average prompt length in a task has a negligible correlation with its performance spread: $r = 0.228$ ($p = 1.4 \times 10^{-7}$) for exact prefix matching metric, and $r = -0.022$ ($p = 0.615$) for option ranking metric, when jointly considering all models and n-shots. Similarly, the standard deviation of the prompt length had negligible correlation with spread: $r = 0.125$ ($p = 0.004$) for exact prefix matching, and $r = -0.099$ ($p = 0.024$) for option ranking metric. When considering each model individually, only LLaMA-2-7B with exact prefix matching showed a correlation $|r| > 0.5$, with the average prompt length having a correlation $r = 0.559$ $p = 6.86 \times 10^{-10}$. All other settings had $|r| < 0.36$.

A.2.3 PCA Examples

Section 2.4.4 systematically analyzes whether we can predict the prompt format that generated a given pre-softmax activation layer (i.e., prompt embeddings) by using solely its top-n principal components. Figure A.24 shows the top two principal components for two different tasks where all 10 formats considered are easily identifiable solely with a prompt embedding's top two principal compoenents.

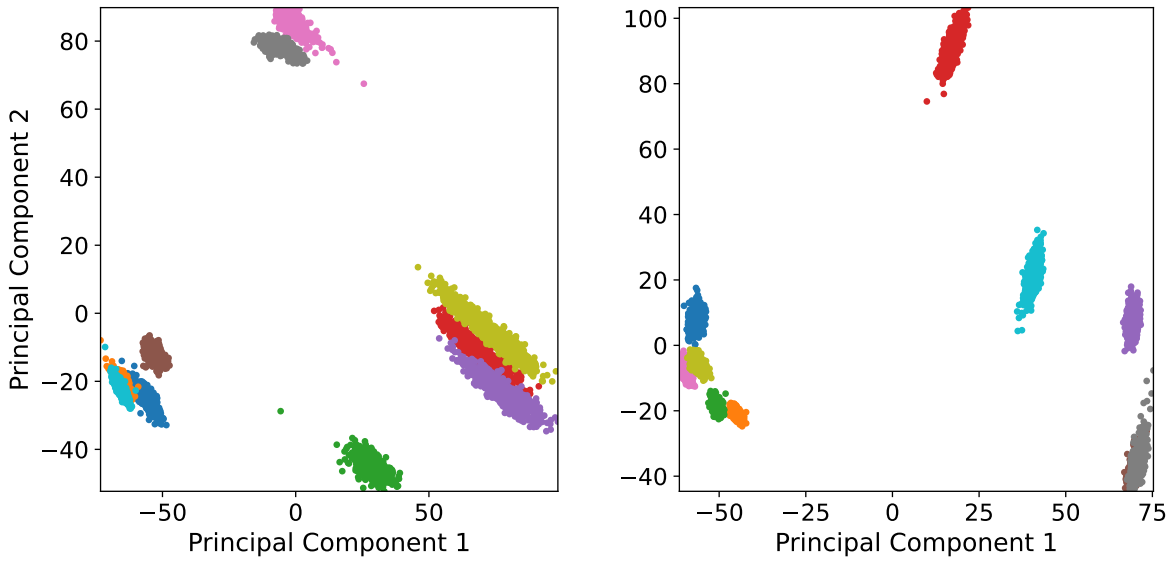


Figure A.24: Plot of the top two principal components of the last decoder layer of the prompt, as a representation of the output probability distribution. Two different tasks shown, with each prompt format shown in a different color.

A.2.4 Notable Features

As discussed in Section 2.4.3, sometimes the choice of a constant may lead to significantly different accuracy ranges. Figures A.25,A.26, and A.27 show all strongly dissimilar choices of constants found on any given task, across 53 Super Natural-Instructions tasks, and on both accuracy metrics considered throughout the work. As can be appreciated, choices of constants do not consistently predict performance in isolation.

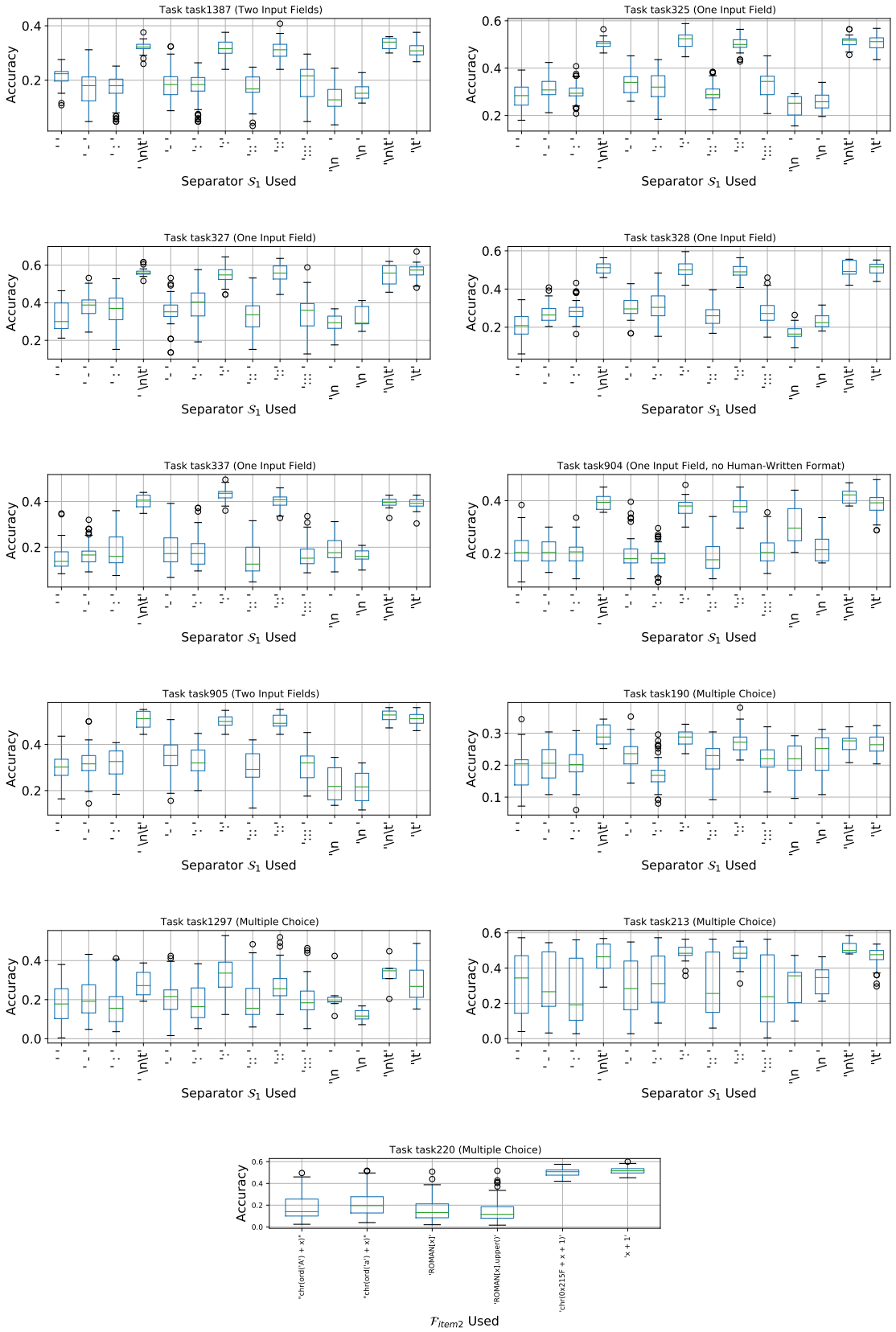


Figure A.25: Variance by feature value for Llama-7B 1-shot. Evaluated 31 tasks with 500 formats each, and only plots with a significant difference between values are shown. Exact prefix matching accuracy metric.

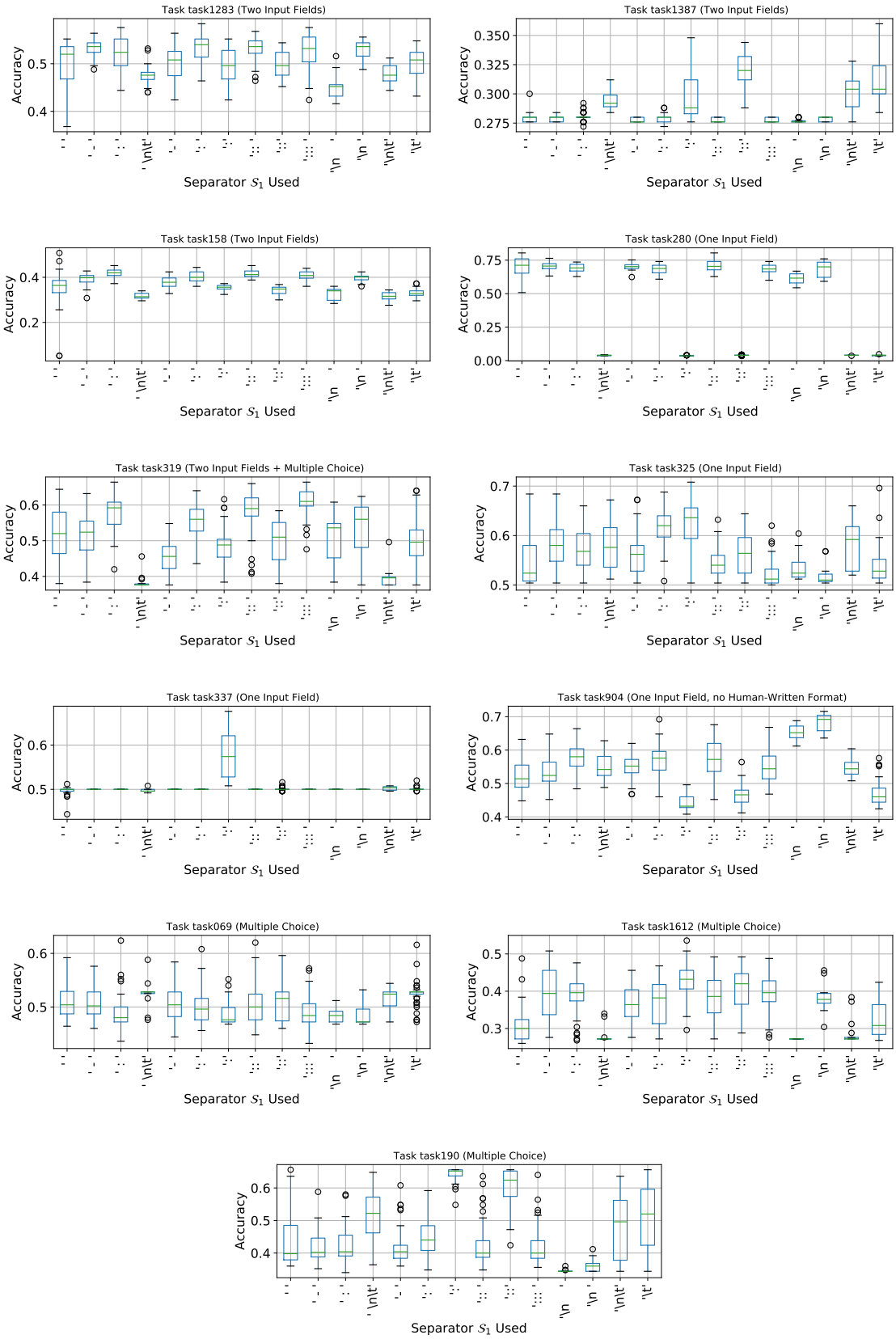


Figure A.26: Variance by feature value for Llama-7B 1-shot. Evaluated 31 tasks with 500 formats each, and only plots with strongly significant differences are shown. Option ranking accuracy evaluation metric. Only $\mathcal{S}_1$ boxplots are shown here, see Figure A.27 for all others.

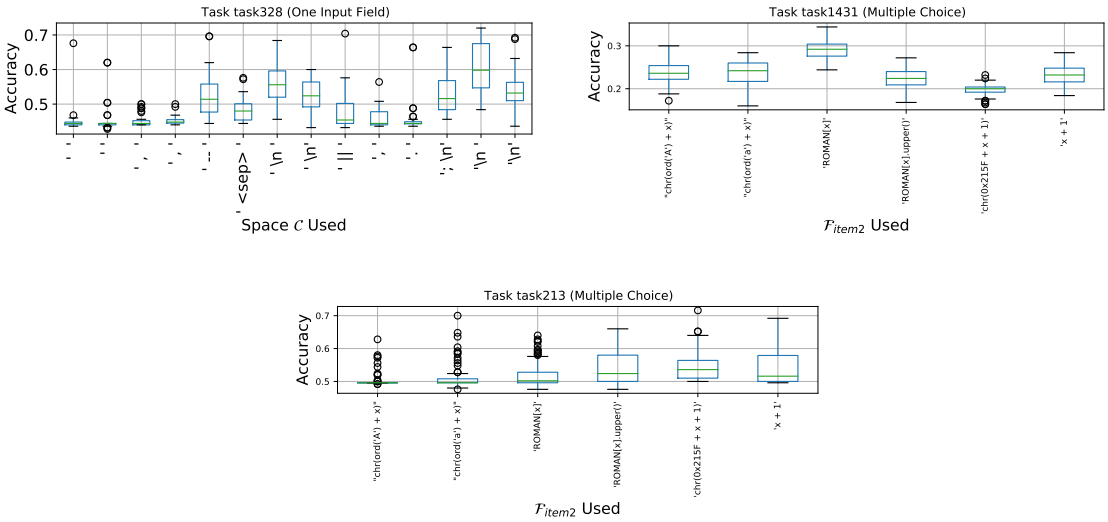


Figure A.27: Variance by feature value for Llama-7B 1-shot. Evaluated 31 tasks with 500 formats each, and only plots with strongly significant differences are shown. Option ranking accuracy evaluation metric. See Figure A.26 for more plots.

A.2.5 Thompson Sampling Results

Task	Model	Best Format	Worst Format	Best Acc	Worst Acc
task050	Llama-70B	`sentence {} || question {} || answer {}`	`Sentence\t{}\n Question\t{}\n Answer\t{}`	0.62	0.58
task065*	Llama-70B	`Sentence<1>:: {} Sentence<3>:: {} Sentence<4>:: {} Sentence<5>:: {} \nOption A.: {} Option B.: {} \nAnswer\n {}`	`Sentence I)\t{} \n Sentence III)\t{} \n Sentence IV)\t{} \n Sentence V)\t{} Option\t(I)::{}, Option\t(II)::{} Answer {}`	0.88	0.50
task069	Llama-70B	`BEGINNING::: {} MIDDLE A): {} , MIDDLE B): {} ENDING::: {} ANSWER::: {}`	`BEGINNING\n {}\n MIDDLE <I>:: {} MIDDLE <II>:: {}\n ENDING\n {}\n ANSWER\n {}`	0.85	0.58
task070	Llama-70B	`Beginning::: {}; \nMiddle\t1)\t{}; \nMiddle\t2)\t{}; \nEnding::: {}; \nAnswer::: {}`	`Beginning {} || Middle 1.: {} || Middle 2.: {} || Ending {} || Answer {}`	0.80	0.27
task114	Llama-70B	`sentence: {} <sep>answer: {}`	`Sentence: {}\nAnswer: {}`	0.54	0.51
task1186	Llama-70B	`SYSTEM REFERENCE : {}. ORIGINAL REFERENCE : {}. ANSWER : {}`	`System Reference: {}\nOriginal Reference: {}\nAnswer: {}`	0.56	0.51
task1283	Llama-70B	`system reference: {} \noriginal reference: {} \nanswer: {}`	`SYSTEM REFERENCE\t{}\n ORIGINAL REFERENCE\t{}\n ANSWER\t{}`	0.62	0.50
task1284	Llama-70B	`SYSTEM REFERENCE\t{} || ORIGINAL REFERENCE\t{} || ANSWER\t{}`	`System Reference {} , Original Reference {} , Answer {}`	0.74	0.43
task1297	Llama-70B	`Fact\tI.\t{} , Fact\tII.\t{} <sep>Question: {}; \n1 ) - {}; \n2 ) - {}; \n3 ) - {}; \n4 ) - {}; \n5 ) - {}; \n6 ) - {}; \n7 ) - {}; \n8 ) - {} <sep>Answer: {}`	`fact a )- {}\nfact b )- {} \nquestion: {} || i) : {}ii) : {}iii) : {}iv) : {}v) : {}vi) : {}vii) : {}viii) : {} \nanswer: {}`	0.88	0.40
task133	Llama-70B	`Sentence:{} \nReason:{} \nQuestion:{} \nAnswer:{}`	`Sentence:: {}\n Reason:: {}\n Question:: {}\n Answer:: {}`	0.71	0.59

task1347	Llama-70B	SENTENCE A): {} , SENTENCE B): {} ANSWER::: {}	Sentence i){} Sentence ii){}; \nAnswer\n {}	0.42	0.31
task1387	Llama-70B	Premise::{} \nHypothesis::{} \nAnswer::{}	premise {}\n hypothesis {}\n answer {}	0.52	0.41
task1419	Llama-70B	problem- {} \noptions- \n <1>:: {}, <2>:: {}, <3>:: {}, <4>:: {}, <5>:: {} \nanswer- {}	Problem: {} || Options: <A>{} <sep><B>{} <sep><C>{} <sep><D>{} <sep><E>{} || Answer: {}	0.24	0.23
task1420	Llama-70B	Problem\t{}\nOptions\t i):{} - ii):{} - iii):{} - iv):{} - v):{}\nAnswer\t{}	PROBLEM- {}\nOPTIONS- \n[a]{}\n [b]{}\n [c]{}\n [d]{}\n [e]{}\nANSWER- {}	0.26	0.22
task1421	Llama-70B	PROBLEM:{}, OPTIONS: \na){} - b){} - c){} - d){} - e){}, ANSWER:{}	Problem::: {} Options::: \nI)::{} \nII)::{} \nIII)::{} \nIV)::{} \nV)::{} Answer::: {}	0.28	0.21
task1423	Llama-70B	problem: {} - options: 1):: {}. 2):: {}. 3):: {}. 4):: {}. 5):: {} - answer: {}	PROBLEM:: {}. OPTIONS:: I.- {}\n II.- {}\n III.- {}\n IV.- {}\n V.- {}. ANSWER:: {}	0.25	0.20
task1502	Llama-70B	Input:{} Output:{}	input {} output {}	0.59	0.45
task155	Llama-70B	SENTENCE - {}, ANSWER - {}	Sentence:: {} \nAnswer:: {}	0.36	0.29
task158	Llama-70B	Sentence: {} || Answer: {}	sentence::{} - answer::{}	0.55	0.49
task161	Llama-70B	Sentence\t{}\n Answer\t{}	Sentence\n {} \nAnswer\n {}	0.45	0.43
task1612*	Llama-70B	sentenceI) - {}sentenceII) - {} \n answer::{}	Sentence (a):{}. Sentence (b):{}\nAnswer {}	0.64	0.35
task162	Llama-70B	Sentence\n {}; \nAnswer\n {}	SENTENCE \n\t{}; \nANSWER \n\t{}	0.45	0.40
task163	Llama-70B	SENTENCE:{}, ANSWER:{}	SENTENCE\n {}\nANSWER\n {}	0.47	0.35
task1678	Llama-70B	PROBLEM: {} , OPTIONS: \n[a] {}. [b] {}. [c] {}. [d] {}. [e] {} , ANSWER: {}	PROBLEM \n\t{} \n OPTIONS \n\t [1] {}[2] {}[3] {}[4] {}[5] {} \n ANSWER \n\t{}	0.21	0.21
task1724	Llama-70B	input: {} - output: {}	INPUT::{} , OUTPUT::{}	0.48	0.48
task190	Llama-70B	Sentence a) - {} - Sentence b) - {}\n Answer\n\t{}	Sentence 1.{}\nSentence 2.{}\n Answer {}	0.66	0.35
task213*	Llama-70B	TITLE::: {} \n Sentence I)::: {} , Sentence II)::: {} , Sentence III)::: {} , Sentence IV)::: {} \n CHOICES::: \ni){} \n ii){} \n ANSWER::: {}	TITLE\t{}, sentence (i)\t{} - sentence (ii)\t{} - sentence (iii)\t{} - sentence (iv)\t{}, CHOICES\t \n (I):: {} (II):: {}, ANSWER\t{}	0.99	0.50
task214	Llama-70B	Title: {}\n Sentence I.: {} Sentence II.: {} Sentence III.: {} Sentence IV.: {}\n Choices: \ni.::: {} \nii.::: {}\n Answer: {}	title {} || sentence [1] {} sentence [2] {} sentence [3] {} sentence [4] {} || choices \n I.{}; \nII.{} || answer {}	0.92	0.04
task220	Llama-70B	SentenceI): {}. SentenceII): {}. SentenceIII): {}. SentenceIV): {}. SentenceV): {} , Choices: \n<a>- {}\n <b>- {} , Answer: {}	Sentence <1>: {} - Sentence <2>: {} - Sentence <3>: {} - Sentence <4>: {} - Sentence <5>: {} , Choices: \n I){} - II){} , Answer: {}	0.99	0.83
task279	Llama-70B	Passage:: {} \n Answer:: {}	Passage- {} Answer- {}	0.64	0.49
task280	Llama-70B	Passage:: {} , Answer:: {}	PASSAGE\t{}\n ANSWER\t{}	0.84	0.04
task286	Llama-70B	INPUT- {} , OUTPUT- {}	Input\t{} Output\t{}	0.69	0.51
task296*	Llama-70B	SentenceI):: {}, SentenceII):: {}, SentenceIII):: {}, SentenceIV):: {}, SentenceV):: {}, SentenceVI):: {}, SentenceVII):: {}, SentenceVIII):: {}, SentenceIX):: {}, SentenceX):: {}, A)\t{} , B)\t{}, Answer : {}	Sentence I.: {}; \nSentence II.: {}; \nSentence III.: {}; \nSentence IV.: {}; \nSentence V.: {}; \nSentence VI.: {}; \nSentence VII.: {}; \nSentence VIII.: {}; \nSentence IX.: {}; \nSentence X.: {} \n <i>::{} || <ii>::{} \n Answer\n {}	0.98	0.53

task297	Llama-70B	Sentence [a]:: {}; \nSentence [b]:: {}; \nSentence [c]:: {}; \nSentence [d]:: {}; \nSentence [e]:: {}; \nSentence [f]:: {}; \nSentence [g]:: {}; \nSentence [h]:: {}; \nSentence [i]:: {}; \nSentence [j]:: {}\n (a):: {} (b):: {}\n Answer {}	Sentence1: {} Sentence2: {} Sentence3: {} Sentence4: {} Sentence5: {} Sentence6: {} Sentence7: {} Sentence8: {} Sentence9: {} Sentence10: {} \n (A) {} (B) {} \n Answer: {}	0.51	0.03
task316	Llama-70B	passage - {} \nanswer - {}	PASSAGE::{}\nANSWER::{}	0.51	0.49
task317	Llama-70B	Passage\n {}\n Answer\n {}	PASSAGE\t{}\n ANSWER\t{}	0.83	0.07
task319	Llama-70B	target\n\t{}; \n{}; \nanswer\n\t{}	Target: {} {} Answer: {}	0.77	0.58
task320	Llama-70B	Target: {}; \n{}; \nAnswer: {}	Target\n {} \n {} \n Answer\n {}	0.77	0.58
task322	Llama-70B	COMMENT:{}\n ANSWER:{}	Comment {} - Answer {}	0.77	0.48
task323	Llama-70B	comment \n\t{}\nanswer \n\t{}	Comment {}. Answer {}	0.84	0.58
task325	Llama-70B	COMMENT\t{}; \nANSWER\t{}	Comment: {}, Answer: {}	0.84	0.48
task326	Llama-70B	Comment:: {} Answer:: {}	Comment: {}, Answer: {}	0.58	0.51
task327	Llama-70B	Comment {} <sep>Answer {}	Comment : {} Answer : {}	0.88	0.69
task328	Llama-70B	comment:{} - answer:{}	Comment: {} Answer: {}	0.81	0.52
task335	Llama-70B	post:: {} , answer:: {}	Post: {}\nAnswer: {}	0.52	0.50
task337	Llama-70B	post {} \nanswer {}	post {} answer {}	0.85	0.63
task385	Llama-70B	CONTEXT {} - QUESTION {} - OPTIONS \n1.- {}, 2.- {}, 3.- {} - ANSWER {}	context \n\t{} \nquestion \n\t{} \noptions \n\t 1)::{}. 2)::{}. 3)::{} \nanswer \n\t{}	0.38	0.11
task580	Llama-70B	Context- {}\n Question- {}\n Options- \na) {} - b) {} - c) {}\n Answer- {}	Context {}, Question {}, Options \n I) - {} \nII) - {} \nIII) - {}, Answer {}	0.78	0.40
task607	Llama-70B	INPUT \n\t{} \n OUTPUT \n\t{}	Input : {}. Output : {}	0.68	0.51
task608	Llama-70B	input: {}. output: {}	INPUT \n\t{}\nOUTPUT \n\t{}	0.71	0.48
task609	Llama-70B	Input \n\t{} \n Output \n\t{}	Input : {}, Output : {}	0.71	0.51
task904	Llama-70B	input::: {} , output::: {}	INPUT::{}\n OUTPUT::{}	0.71	0.55
task905	Llama-70B	Tweet::{} || Label::{} || Answer::{}	TWEET: {} \n LABEL: {} \n ANSWER: {}	0.68	0.50
task050	GPT3.5	Sentence\n\t{} \n Question\n\t{} \n Answer\n\t{}	sentence: {} , question: {} , answer: {}	0.67	0.61
task065	GPT3.5	SENTENCEi.::: {} \nSENTENCEiii.::: {} \nSENTENCEiv.::: {} \nSENTENCEv.::: {} \nOPTION [1]\t{}OPTION [2]\t{} \nANSWER\t{}	Sentence 1) : {} , Sentence 3) : {} , Sentence 4) : {} , Sentence 5) : {} <sep>Option (i): {} <sep>Option (ii): {} <sep>Answer- {}	0.82	0.33
task069	GPT3.5	BEGINNING: {}\n MIDDLE\t(I)::: {} \n MIDDLE\t(II)::: {}\n ENDING: {}\n ANSWER: {}	Beginning {} || Middle [a]- {} \n Middle [b]- {} || Ending {} || Answer {}	0.83	0.65
task070*	GPT3.5	Beginning : {}\nMiddle (I): {} - Middle (II): {}\nEnding : {}\nAnswer : {}	Beginning\t{} <sep>Middle i){} || Middle ii){} <sep>Ending\t{} <sep>Answer\t{}	0.70	0.49
task114	GPT3.5	sentence \n\t{}\nanswer \n\t{}	SENTENCE: {}. ANSWER: {}	0.67	0.63
task1186	GPT3.5	system reference : {} , original reference : {} , answer : {}	System Reference: {}\nOriginal Reference: {}\nAnswer: {}	0.53	0.51

task1283	GPT3.5	SYSTEM REFERENCE : {} , ORIGINAL REFERENCE : {} , ANSWER : {}	System Reference\n {} \nOriginal Reference\n {} \nAnswer\n {}	0.57	0.50
task1284	GPT3.5	System Reference \n\t{}\n Original Reference \n\t{}\n Answer \n\t{}	system reference::: {} - original reference::: {} - answer::: {}	0.63	0.57
task1297*	GPT3.5	Fact<1>: {} - Fact<2>: {}; \nQuestion::{}\nI):: {} II):: {} III):: {} IV):: {} V):: {} VI):: {} VII):: {} VIII):: {}; \nAnswer::{}	fact\ta. : {} || fact\tb. : {}\n question : {} \na. - {}b. - {}c. - {}d. - {}e. - {}f. - {}g. - {}h. - {}\n answer : {}	0.84	0.72
task133	GPT3.5	Sentence::{} - Reason::{} - Question::{} - Answer::{}	Sentence - {}; \nReason - {}; \nQuestion - {}; \nAnswer - {}	0.69	0.64
task1347*	GPT3.5	SentenceI.:: {}\n SentenceII.:: {} Answer\t{}	Sentence (I){} Sentence (II){} \nAnswer::{}	0.46	0.42
task1387	GPT3.5	Premise:{} , Hypothesis:{} , Answer:{}	PREMISE:: {}; \nHYPOTHESIS:: {}; \nANSWER:: {}	0.47	0.44
task1419	GPT3.5	Problem - {} || Options - \n[A]\t{}. [B]\t{}. [C]\t{}. [D]\t{}. [E]\t{} || Answer - {}	PROBLEM- {}\nOPTIONS- 1) - {} - 2) - {} - 3) - {} - 4) - {} - 5) - {}\nANSWER- {}	0.24	0.20
task1420*	GPT3.5	problem- {} \n options- (I) - {}(II) - {}(III) - {}(IV) - {}(V) - {} \n answer- {}	Problem:: {} - Options:: \n[i]: {}; \n[ii]: {}; \n[iii]: {}; \n[iv]: {}; \n[v]: {} - Answer:: {}	0.24	0.09
task1421	GPT3.5	Problem \n\t{}\n Options \n\t \nA)::{} - B)::{} - C)::{} - D)::{} - E)::{}\n Answer \n\t{}	PROBLEM::{}, OPTIONS:: \n[a]: {} \n[b]: {} \n[c]: {} \n[d]: {} \n[e]: {}, ANSWER::{}	0.30	0.01
task1423	GPT3.5	PROBLEM: {}; \nOPTIONS: \n a) {} \n b) {} \n c) {} \n d) {} \n e) {}; \nANSWER: {}	Problem:: {} - Options:: \n[i]: {}; \n[ii]: {}; \n[iii]: {}; \n[iv]: {}; \n[v]: {} - Answer:: {}	0.30	0.10
task1502	GPT3.5	input\n {}\noutput\n {}	input::{} - output::{}	0.54	0.48
task155	GPT3.5	SENTENCE::{} - ANSWER::{}	SENTENCE {} ANSWER {}	0.48	0.40
task158	GPT3.5	Sentence:{} Answer:{}	SENTENCE\t{} , ANSWER\t{}	0.63	0.60
task161	GPT3.5	sentence\t{} answer\t{}	Sentence- {}\nAnswer- {}	0.40	0.31
task1612	GPT3.5	sentence 1): {} || sentence 2): {}, answer::{}	Sentence I) : {} \nSentence II) : {} , Answer {}	0.66	0.54
task162	GPT3.5	Sentence- {} - Answer- {}	SENTENCE\t{}; \nANSWER\t{}	0.37	0.31
task163	GPT3.5	Sentence- {}; \nAnswer- {}	sentence: {}\nanswer: {}	0.43	0.39
task1678*	GPT3.5	Problem - {} Options - \n (I) {} <sep>(II) {} <sep>(III) {} <sep>(IV) {} <sep>(V) {} Answer - {}	PROBLEM::{} \n OPTIONS:: \n[I] {} , [II] {} , [III] {} , [IV] {} , [V] {} \n ANSWER::{}	0.24	0.09
task1724	GPT3.5	INPUT::{} || OUTPUT::{}	INPUT: {} OUTPUT: {}	0.63	0.57
task190	GPT3.5	Sentence (I) - {}; \nSentence (II) - {}; \nAnswer {}	Sentence<1>: {}\nSentence<2>: {}, Answer : {}	0.29	0.16
task213	GPT3.5	Title: {} <sep>Sentence <1>: {} || Sentence <2>: {} || Sentence <3>: {} || Sentence <4>: {} <sep>Choices: \n<i>- {} <sep><ii>- {} <sep>Answer: {}	TITLE:{}, sentence\t[a]:{} - sentence\t[b]:{} - sentence\t[c]:{} - sentence\t[d]:{}, CHOICES: \n[i]:{} [ii]:{}, ANSWER:{}	0.98	0.81
task214*	GPT3.5	title: {} \nSentencei): {}, Sentenceii): {}, Sentenceiii): {}, Sentenceiv): {} \nchoices: \n<I>- {} , <II>- {} \nanswer: {}	Title::{} , Sentence\ta)::{} , Sentence\tb)::{} , Sentence\tc)::{} , Sentence\td)::{} , Choices:: [i]. {}[ii]. {} , Answer::{}	0.97	0.41
task220	GPT3.5	Sentence [1]: {} \n Sentence [2]: {} \n Sentence [3]: {} \n Sentence [4]: {} \n Sentence [5]: {} - Choices: \n <A>::{}\n <B>::{} - Answer: {}	Sentence\ti.: {} || Sentence\tii.: {} || Sentence\tiii.: {} || Sentence\tiv.: {} || Sentence\tv.: {} Choices: \n[i]:{} || [ii]:{} Answer: {}	0.98	0.79
task279	GPT3.5	passage {} , answer {}	Passage:{}, Answer:{}	0.58	0.56
task280	GPT3.5	Passage - {} <sep>Answer - {}	Passage {} - Answer {}	0.85	0.80
task286	GPT3.5	Input- {}\n Output- {}	INPUT:: {} OUTPUT:: {}	0.72	0.69

task296	GPT3.5	Sentence a) - {} \nSentence b) - {} \nSentence c) - {} \nSentence d) - {} \nSentence e) - {} \nSentence f) - {} \nSentence g) - {} \nSentence h) - {} \nSentence i) - {} \nSentence j) - {} \n<A>- {}, <B>- {} \nAnswer::: {}	Sentence\t(I)- {} - Sentence\t(II)- {} - Sentence\t(III)- {} - Sentence\t(IV)- {} - Sentence\t(V)- {} - Sentence\t(VI)- {} - Sentence\t(VII)- {} - Sentence\t(VIII)- {} - Sentence\t(IX)- {} - Sentence\t(X)- {}, I)::{}, II)::{}, Answer:{}	0.95	0.68
task297	GPT3.5	Sentence (A): {}; \nSentence (B): {}; \nSentence (C): {}; \nSentence (D): {}; \nSentence (E): {}; \nSentence (F): {}; \nSentence (G): {}; \nSentence (H): {}; \nSentence (I): {}; \nSentence (J): {}; \n<I>::{} <sep><II>::{}; \nAnswer {}	SENTENCE\t(A) : {}; \nSENTENCE\t(B) : {}; \nSENTENCE\t(C) : {}; \nSENTENCE\t(D) : {}; \nSENTENCE\t(E) : {}; \nSENTENCE\t(F) : {}; \nSENTENCE\t(G) : {}; \nSENTENCE\t(H) : {}; \nSENTENCE\t(I) : {}; \nSENTENCE\t(J) : {} - I) : {} II) : {} - ANSWER: {}	0.36	0.06
task316	GPT3.5	passage\n\t{} \nanswer\n\t{}	Passage- {} Answer- {}	0.49	0.48
task317	GPT3.5	passage: {} answer: {}	passage {} answer {}	0.75	0.70
task319	GPT3.5	TARGET:: {} \n{} \nANSWER:: {}	Target: {}\n{}\nAnswer: {}	0.66	0.62
task320	GPT3.5	Target::: {} \n{} \nAnswer::: {}	target {} - {} - answer {}	0.73	0.68
task322	GPT3.5	Comment: {} - Answer: {}	comment::{} answer::{}	0.84	0.83
task323	GPT3.5	COMMENT {} <sep>ANSWER {}	comment - {}\nanswer - {}	0.73	0.63
task325	GPT3.5	comment {}; \nanswer {}	Comment\n {}\n Answer\n {}	0.81	0.74
task326	GPT3.5	comment : {} , answer : {}	Comment- {} Answer- {}	0.67	0.66
task327	GPT3.5	comment:{} \nanswer:{}	Comment - {} || Answer - {}	0.86	0.82
task328	GPT3.5	Comment - {}. Answer - {}	COMMENT {} <sep>ANSWER {}	0.77	0.72
task335	GPT3.5	Post\n\t{} \nAnswer\n\t{}	post - {} - answer - {}	0.37	0.20
task337	GPT3.5	Post::{}; \nAnswer::{}	Post {}, Answer {}	0.57	0.51
task385	GPT3.5	Context:: {}; \nQuestion:: {}; \nOptions:: a)::: {}\n b)::: {}\n c)::: {}; \nAnswer:: {}	Context:{}. Question:{}. Options:\n[i] {}[ii] {}[iii] {}. Answer:{}	0.46	0.10
task580	GPT3.5	Context:: {} || Question:: {} || Options:: \n A)- {} B)- {} C)- {} || Answer:: {}	CONTEXT:: {} - QUESTION:: {} - OPTIONS:: \n[I] - {}. [II] - {}. [III] - {} - ANSWER:: {}	0.74	0.65
task607	GPT3.5	input {} || output {}	INPUT:: {}\nOUTPUT:: {}	0.70	0.68
task608	GPT3.5	Input:: {} Output:: {}	Input\n {} \nOutput\n {}	0.60	0.52
task609	GPT3.5	INPUT\n {} \nOUTPUT\n {}	Input {}\nOutput {}	0.70	0.68
task904	GPT3.5	INPUT : {}\nOUTPUT : {}	Input:: {}, Output:: {}	0.66	0.61
task905	GPT3.5	Tweet:{} , Label:{} , Answer:{}	TWEET {} || LABEL {} || ANSWER {}	0.72	0.63

Table A.4: Best and worst formats found by using Thompson Sampling on 320 formats on Llama-70B and GPT3.5 , $E = 40000$, $B = 10$. Tasks marked with an asterisk* indicate a format where the Roman numerals used correspond to its Unicode characters (starting at 0x215F), visualized as I, II, III, IV, ...

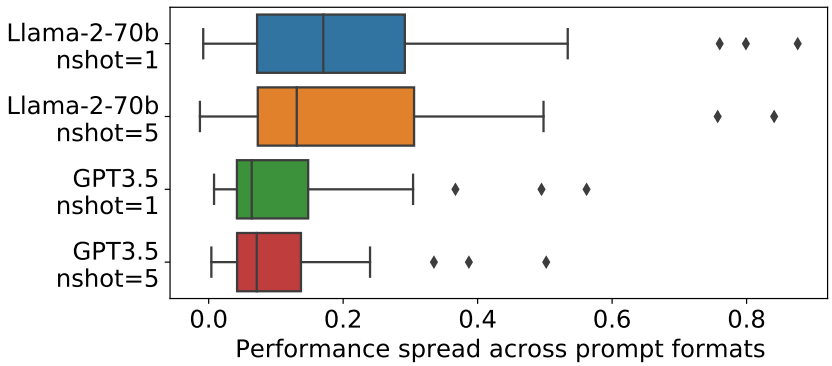


Figure A.28: Spreads found using Thompson Sampling for 320 formats, budget of 40000 evaluations. LLaMA-2-70b was evaluated with option ranking, and GPT3.5 with prefix matching given that we cannot access all logits.

Appendix B

Faith and Fate: Additional Experiments and Proofs

B.1 Compositional Tasks

B.1.1 Multiplication

Data Construction We exhaustively generate multiplication problems as question-answer pairs (e.g., Q: "What is 4 times 32?" A: "128"). We focus on multiplications of two numbers $x = (x_1, x_2, \ldots, x_k)$ and $y = (y_1, y_2, \ldots, y_k)$ where each number can have up to k digits, amounting to $9 \times 10^{(k-1)}$ combinations per each number. We set k to 5 in our experiments. Figure B.1 showcases an example prompt for performing few-shot learning without the inclusion of a scratchpad, while Figure B.2 demonstrates an example prompt using a scratchpad. Throughout our experimentation, we explored various versions of the scratchpad, ranging from verbose and detailed to more concise alternatives. Among these variations, the scratchpad version depicted in Figure B.2 ultimately produced the most favorable outcomes. Listing B.1.1 shows the Python code for solving the task.

```
To multiply two numbers, start by multiplying the rightmost digit of the
   multiplicand by each digit of the multiplier, writing down the products
    and carrying over any remainders.  Repeat this process for each digit
   of the multiplicand, and then add up all the partial products to obtain
    the final result.

Questions: what's 22 times 2? Answer 44.
```

Figure B.1: Example prompt for the multiplication task used for the few-shot setting.

```
def multiply(x, y):
    summands = [0] * len(y)
    for i in range(len(y) - 1, -1, -1):
        digits = [0] * len(x)
        carry = 0
        for j in range(len(x) - 1, -1, -1):
            t = x[j] * y[i]
            t += carry
            carry = t // 10
            digits[j] = t % 10
        digits.insert(0, carry)
        summands[i] = sum(digits[-k] * (10 ** (k - 1)) for k in
            range(1, len(digits) + 1))

    product = sum(summands[-i] * (10 ** (i - 1)) for i in
        range(1, len(y) + 1))
    return product
```

```
Question: What is 35 times 90?


Scratchpad: Let's perform the multiplication step by step:

Let's multiply 35 by the digit in the ones place of 90, which is 0.

1. Multiply 0 by the digit in the ones place of 35, which is 5. This gives
    5 x 0 = 0. Write down the result 0.
2. Multiply 0 by the digit in the tens place of 35, which is 3. This gives
    3 x 0 = 0. Write down the result 0.
3. The partial product for this step is A=0 which is the concatenation of
   the digits we found in each step.

Now, let's multiply 35 by the digit in the tens place of 90, which is 9.

4. Multiply 9 by the digit in the ones place of 35, which is 5. This gives
    5 x 9 = 45. Write down the result 5 and carry over the 4 to the next
   step.
5. Multiply 9 by the digit in the tens place of 35, which is 3. Add the
   carryover from the previous step to account for this. This gives (3 x
   9) + 4 = 31. Write down the result 31.
6. The partial product for this step is B=315 which is the concatenation
   of the digits we found in each step.

Now, let's sum the 2 partial products A and B, and take into account the
   position of each digit: A=0 (from multiplication by 0) and B=315 (from
   multiplication by 9 but shifted one place to the left, so it becomes
   3150). The final answer is 0 x 1 + 315 x 10 = 0 + 3150 = 3150.
```

Figure B.2: A sample scratchpad for the multiplication task.

B.1.2 Dynamic Programming Problem

B.1.2.1 Solution to this problem

Let $a = [a_1, \ldots, a_n]$ be an input. Let dp_i be the maximum sum of a subsequence that does not include adjacent elements, when considering only the elements of the input from the i-th position onwards.

Trivially, $dp_n = \max(a_n, 0)$ since we only want to choose a number if it is non-negative. Moreover, $dp_{n-1} = \max(a_n, a_{n-1}, 0)$ since we cannot choose adjacent numbers.

For any given dp_i with $i \leq n-2$, we can express it in terms of dp_{i+1} and dp_{i+2}. Concretely, the maximum sum of a subsequence starting at position i may or may not include the element in the i-th position, a_i. If the subsequence includes a_i, then the maximum sum is $a_i + dp_{i+2}$, since using a_i blocks us from using the next element. If the subsequence does not include a_i, then its sum is dp_{i+1}. Moreover, the answer may never be less than zero, because otherwise we would select the empty sequence[1]. In summary,

$$dp_i = \max(dp_{i+1},\ a_i + dp_{i+2},\ 0)$$

We now have a recursion with its base cases $dp_n = \max(a_n, 0)$ and $dp_{n-1} = \max(a_n, a_{n-1}, 0)$, and we can therefore compute all values in $O(n)$. It now only rests to reconstruct the lexicographically smallest subsequence that maximizes the desired sum, based solely on the computed dp values.

Starting from dp_1 and iterating sequentially through dp_{n-2}, we choose an item if and only if $dp_i = a_i + dp_{i+2}$ (that is, the maximum sum comes from choosing the current element) and we have not chosen the previous element. This helps disambiguate cases where choosing or not choosing a_i yields the same sum, but possibly only one of those will not incur in choosing adjacent numbers. Similarly, for positions $i = n-1$ and $i = n$ we choose the element if $dp_i = a_i$ (that is, choosing the element yields the maximum sum) and we have not chosen the immediately previous element. See an example Python solution in B.1.2.1.

```
Given a sequence of integers, find a subsequence with the highest sum,
   such that no two numbers in the subsequence are adjacent in the
   original sequence.

Output a list with "1" for chosen numbers and "2" for unchosen ones. If
   multiple solutions exist, select the lexicographically smallest. input
   = [3, 2, 1, 5, 2].
```

Figure B.3: Example prompt for the DP task, used for zero-shot and few-shot settings.

[1] We don't need to explicitly check for this since $dp_n \geq 0$. However, we include the condition to ease the scratchpad logic.

```
def maximum_sum_nonadjacent_subsequence(arr):

    N = len(arr)
    dp = [0 for _ in range(N)]

    dp[N - 1] = max(arr[N - 1], 0)
    dp[N - 2] = max(max(arr[N - 1], arr[N - 2]), 0)

    for i in range(N - 3, -1, -1):
        dp[i] = max(max(dp[i + 1], arr[i] + dp[i + 2]), 0)

    # reconstruct the answer with a fixed-size graph
    result = []
    can_use_next_item = True

    for i in range(N - 2):
        if dp[i] == arr[i] + dp[i + 2] and can_use_next_item:
            result.append(1)
            can_use_next_item = False
        else:
            result.append(2)
            can_use_next_item = True

    if dp[N - 2] == arr[N - 2] and can_use_next_item:
        result.append(1)
        can_use_next_item = False
    else:
        result.append(2)
        can_use_next_item = True

    if dp[N - 1] == arr[N - 1] and can_use_next_item:
        result.append(1)
    else:
        result.append(2)

    return result
```

Data Construction We exhaustively generate data for this DP task. For question-answer setting, we include a thorough explanation of the task before asking to generate a solution (see Figure B.3). We use all lists up to 5 elements as training, and we consider only lists where elements are in the range $[-5, 5]$ (giving a total of 11^n lists for an input list of size n). For out-of-domain evaluation, we use lists of sizes 6 to 10 inclusive. Example scratchpads and zero-shot prompts are shown in Figure B.4 and B.3 respectively. The scratchpad is generated automatically through templates. We considered five exemplars for the few-shot

```
Question: Let's solve input = [3, 2, 1, 5, 2].


Scratchpad: dp[4] = max(input[4], 0) = max(2, 0) = 2
dp[3] = max(input[3], input[4], 0) = max(5, 2, 0) = 5
dp[2] = max(dp[3], input[2] + dp[4], 0) = max(5, 1 + 2, 0) = 5
dp[1] = max(dp[2], input[1] + dp[3], 0) = max(5, 2 + 5, 0) = 7
dp[0] = max(dp[1], input[0] + dp[2], 0) = max(7, 3 + 5, 0) = 8

Finally, we reconstruct the lexicographically smallest subsequence that
   fulfills the task objective by selecting numbers as follows. We store
   the result on a list named "output".

Let can_use_next_item = True.
Since dp[0] == input[0] + dp[2] (8 == 3 + 5) and can_use_next_item == True
   , we store output[0] = 1. We update can_use_next_item = False.
Since dp[1] != input[1] + dp[3] (7 != 2 + 5) or can_use_next_item == False
   , we store output[1] = 2. We update can_use_next_item = True.
Since dp[2] != input[2] + dp[4] (5 != 1 + 2) or can_use_next_item == False
   , we store output[2] = 2. We update can_use_next_item = True.
Since dp[3] == input[3] (5 == 5) and can_use_next_item == True, we store
   output[3] = 1. We update can_use_next_item = False.
Since dp[4] != input[4] (2 != 2) or can_use_next_item == False, we store
   output[4] = 2.

Reconstructing all together, output=[1, 2, 2, 1, 2].
```

Figure B.4: A sample scratchpad for the DP task used for fine-tuning with few-shot settings.

setup.

B.2 Experimental Setups & Empirical Results

B.2.1 Models

For our experiments, we evaluate the performance of two LLMs: GPT4 (`gpt-4`) and GPT3 (`text-davinci-003`). The evaluations were conducted from January 2023 to May 2023 using the OpenAI API. We perform fine-tuning on GPT3 (`text-davinci-003`) for the two tasks, observing faster convergence when training on question-scratchpad pairs rather than question-answer pairs. For question-answer pairs fine-tuning, we train separately the model for $\{14, 4\}$ epochs for multiplication and DP respectively, saving the best model based on the validation set. Regarding training on question-scratchpad pairs, we train the model for $\{16, 2\}$ epochs for multiplication and DP. The batch size is set to approximately 0.2% of the number of examples in the training set. Generally, we observe that larger batch sizes tend to yield better results for larger datasets. For the learning rate multiplier, we experiment with values ranging from 0.02 to 0.2 to determine the optimal setting for achieving the best results and chose 0.2. During inference, we set nucleus sampling p to 0.7 and temperature to 1. For each task, we evaluate the performance of each model on 500 test examples.

Dziri*, Lu*, Sclar* et al. 2023 also evaluate the performance of 4 other LLMs: ChatGPT (`GPT3.5-turbo`) (OpenAI, 2022), FlanT5 (Chung et al., 2024) and LLaMA (Touvron et al., 2023b), showing the same trends across all models.

B.3 Surface Patterns

B.3.1 Relative Information Gain Predictions for Multiplication

		Relative Information Gain			
Input variable	Output variable	2x2	3x3	4x4	5x5
x_n	z_{2n}	0.223	0.223	0.223	0.223
y_n	z_{2n}	0.223	0.223	0.223	0.223
x_1	z_1	0.198	0.199	0.199	0.199
y_1	z_1	0.198	0.199	0.199	0.199
$x_n\ y_n$	z_{2n}	1.000	1.000	1.000	1.000
$x_{n-1}\ x_n$	z_{2n}	0.223	0.223	0.223	0.223
$y_{n-1}\ y_n$	z_{2n}	0.223	0.223	0.223	0.223
$x_n\ y_n$	z_{2n-1}	0.110	0.101	0.101	0.101
$y_{n-1}\ y_n$	z_{2n-1}	0.032	0.036	0.036	0.036
$x_{n-1}\ x_n$	z_{2n-1}	0.032	0.036	0.036	0.036
$x_{n-1}\ y_{n-1}$	z_{2n-1}	0.018	0.025	0.025	0.025
$x_1\ y_1$	z_2	0.099	0.088	0.088	0.088
$x_2\ y_2$	z_2	0.025	0.016	0.016	0.016
$x_1\ y_1$	z_1	0.788	0.792	0.793	0.793
$y_1\ y_2$	z_1	0.213	0.211	0.211	0.211
$x_1\ x_2$	z_1	0.213	0.211	0.211	0.211

Table B.1: **Highest Relative Information Gain Elements and Pairs of Elements**, for multiplications between $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$, with $2 \leq n \leq 5$. We define $z := x \cdot y$, which will always have size $2n$ (with possibly a leading zero). z_{2n} denotes the least-significant digit of z, and z_1 denotes the left-most digit. Only (input, output) pairs above 0.01 are shown. Note that since multiplication is commutative, several pairs of input variables (e.g. a_0 and b_0) exhibit the same relative information gain.

B.3.2 Empirical Surface Pattern Analysis for Multiplication with GPT4, ChatGPT and GPT3

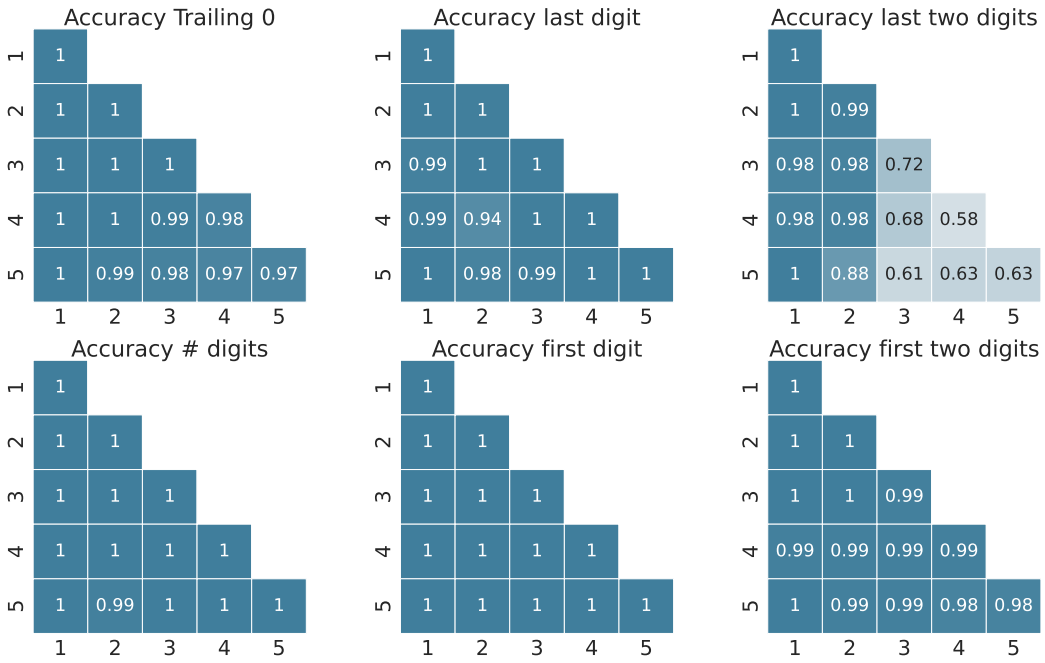


Figure B.5: GPT4 zero-shot accuracy in predicting partially correct responses. This evidences surface pattern learning, since the accuracy of full answer prediction is significantly lower–and often near zero (see Figure 3.2). Specifically, 'accuracy trailing zeros' pertains to accurately predicting the number of zeros in the output number, which is known to be relatively easy to predict based on arithmetic calculations.

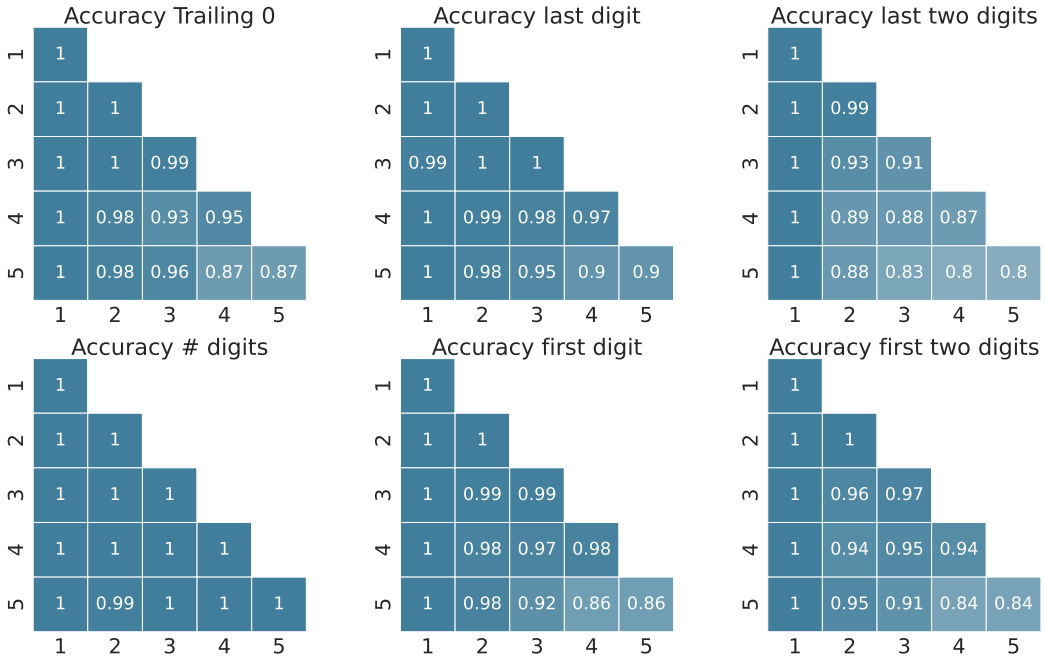


Figure B.6: ChatGPT zero-shot accuracy in predicting partially correct responses. We observe the same trend for GPT3 predictions.

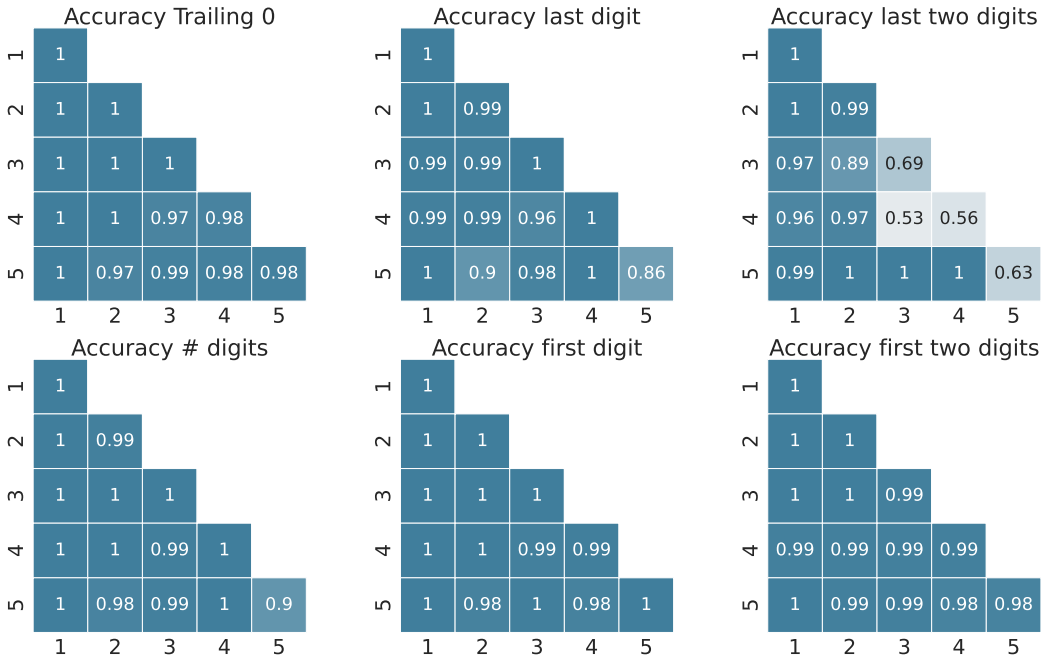


Figure B.7: GPT4 five-shot accuracy in predicting partially correct responses. We observe the same trend for ChatGPT, GPT3 few-shot predictions.

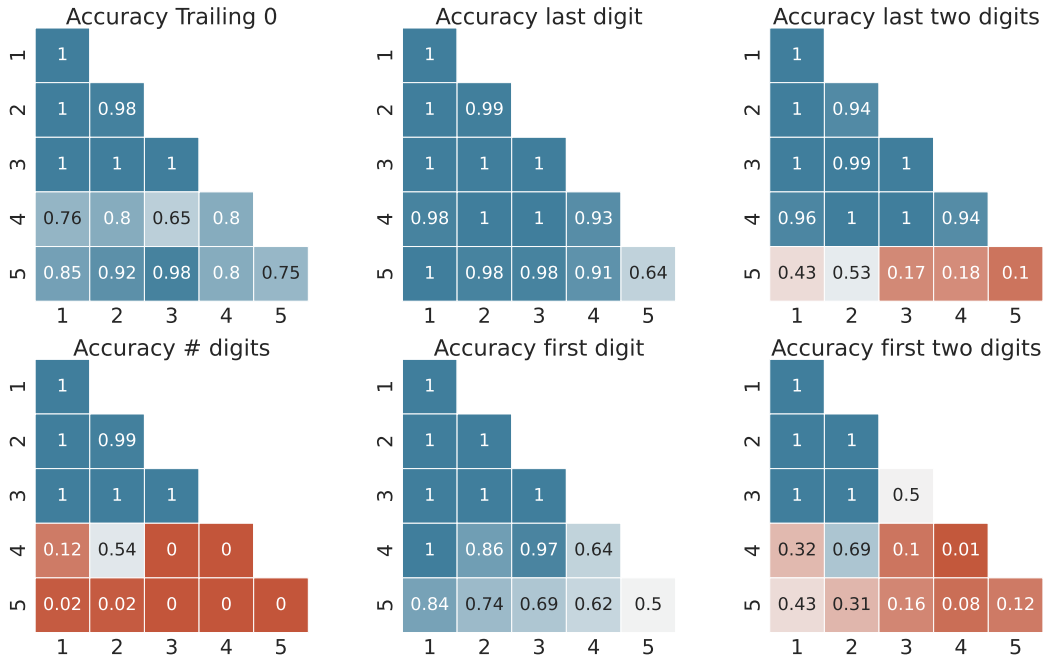


Figure B.8: GPT3 finetuned on question-scratchpad pairs. Accuracy of predicting partially correct responses.

B.3.3 Relative Information Gain Predictions for Dynamic Programming Task

Let a_i be the i-th element of the input sequence, and let o_i be the i-th element of the output sequence. As shown in Table B.2, a_i is a good predictor of o_i, and this is especially true for a_1 and a_{n-1}, the first and last elements of the sequence. This matches the task intuition, since one would never pick an element $a_i < 0$ and decrease the final sum (one may pick $a_i = 0$ if it makes a lexicographically smaller output sequence).

a_i weakly helps to predict its neighbors. The only case of this behavior with RelativeIG>0.1 is at the start of the sequence, where the first element helps predict the value of the second. This again matches intuition, since a very high a_1 indicates that with high probability o_2 will not be selected for the final subsequence.

Similar behaviors, but with higher relative information gains overall, are observed when analyzing triples of consecutive elements in the list. Table B.3 shows that o_i is highly predicted by (a_{i-1}, a_i, a_{i+1}). Moreover, o_i is highly predicted by both (a_{i-2}, a_{i-1}, a_i) and (a_i, a_{i+1}, a_{i+2}), with the former generally having higher scores than the latter. This again matches the task intuitions, since the value of the neighbors helps determine whether to select a number for the subsequence; and asking for the lexicographically smallest sequence biases the output subsequence to care more about the previous numbers rather than the following ones. We believe that this last point is the cause of the weakly predictive power of $(a_{i-3}, a_{i-2}, a_{i-1})$ to predict o_i; whereas $(a_{i+1}, a_{i+2}, a_{i+3})$ is not shown, since all the relative information gain values were below 0.1.

		Relative Information Gain for each problem size								
Input variable	Output variable	2	3	4	5	6	7	8	9	10
a_1	o_2	0.15	0.13	0.14	0.14	0.14	0.14	0.14	0.14	0.14
a_1	o_1	0.64	0.71	0.69	0.69	0.69	0.69	0.69	0.69	0.69
a_2	o_2	0.53	0.42	0.45	0.44	0.45	0.44	0.44	0.45	0.44
a_3	o_3		0.64	0.49	0.53	0.52	0.52	0.52	0.52	0.52
a_4	o_4			0.60	0.46	0.50	0.49	0.49	0.49	0.49
a_5	o_5				0.62	0.47	0.51	0.50	0.50	0.50
a_6	o_6					0.61	0.47	0.51	0.49	0.50
a_7	o_7						0.61	0.47	0.51	0.50
a_8	o_8							0.61	0.47	0.51
a_9	o_9								0.61	0.47
a_{10}	o_{10}									0.61
a_{n-1}	o_{n-1}		0.64	0.60	0.62	0.61	0.61	0.61	0.61	0.61
a_{n-2}	o_{n-2}				0.46	0.47	0.47	0.47	0.47	0.47
a_{n-3}	o_{n-3}						0.51	0.51	0.51	0.51
a_{n-4}	o_{n-4}								0.49	0.50

Table B.2: **Highest Relative Information Gain Elements**, for DP problems of size $2 \leq n \leq 10$.
We only show the (input, output) pairs where at least three problem sizes have RelativeIG>0, and at least one with RelativeIG>0.1. a_{n-1} refers to the last element of the sequence, regardless of its actual id in the sequence.

B.3.4 Empirical Surface Pattern Results for Dynamic Programming Task

We observe that all analyzed models match the Relative Information Gain prediction that o_1 (whether the first element goes into the output sequence or not) should be the easiest value to predict (see Figures B.9, B.10, and B.11). However, since GPT3 often predicts shorter output sequences than the required size, the analysis of the predictive power of o_{n-1} is only done for GPT4. In GPT4, we observe that o_{n-1} is among the easiest values to predict as expected by Relative Information Gain.

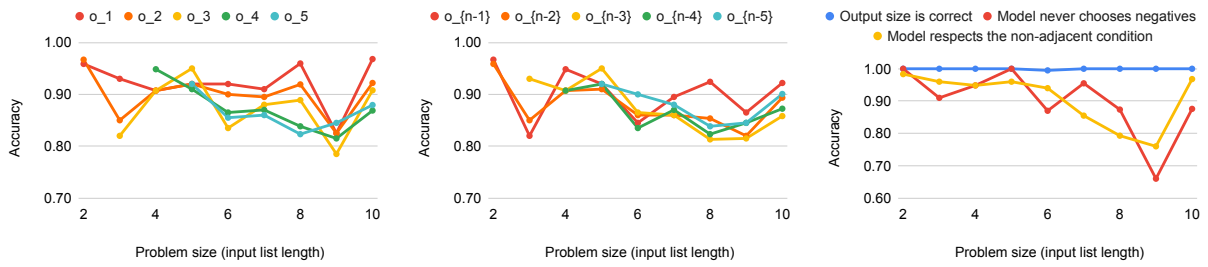


Figure B.9: GPT4 five-shot with scratchpad accuracy in predicting output elements o_i in the DP task. All o_i are predicted with high accuracy with o_1 and o_{n-1} being consistently among the highest. These observations go in line with the Relative Information Gain prediction.

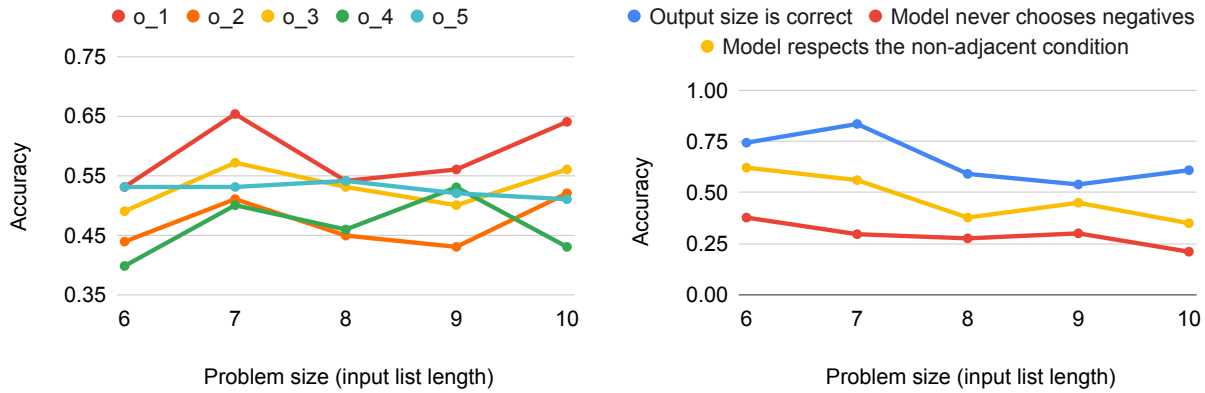


Figure B.10: GPT3 few-shot without scratchpad accuracy in predicting output elements o_i in the DP task. As predicted by Relative Information Gain, the model predicts o_1 correctly with the highest probability. However, because GPT3 often does not produce the correct output size, it hinders us from analyzing o_{n-1}.

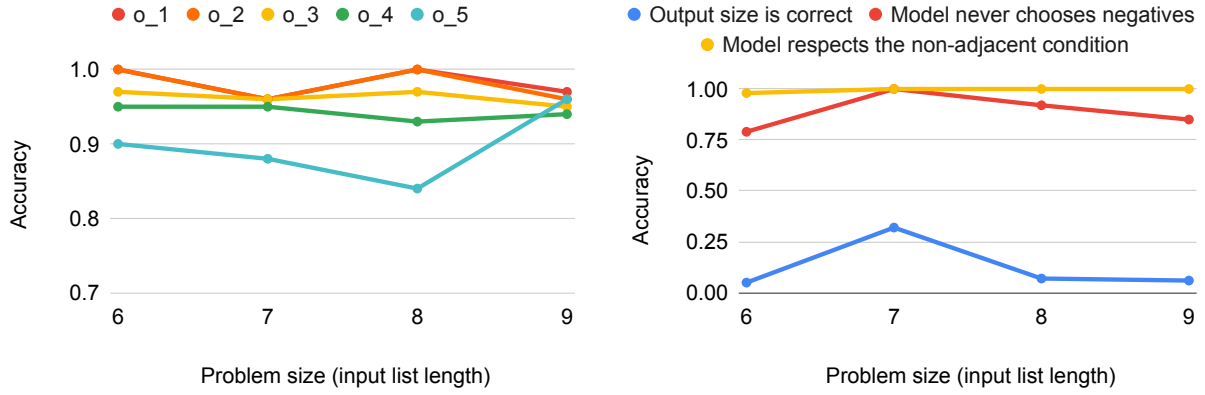


Figure B.11: GPT3 fine-tuned without scratchpad accuracy in predicting output elements o_i in the DP task. As predicted by Relative Information Gain, the model predicts o_1 correctly with the highest probability. However, because GPT3 often does not produce the correct output size, it hinders us from analyzing o_{n-1}.

		Relative Information Gain for each problem size							
Input variable	Output variable	3	4	5	6	7	8	9	10
$a_{n-3}\ a_{n-2}\ a_{n-1}$	o_{n-1}					0.95	0.95	0.95	0.95
$a_{n-3}\ a_{n-2}\ a_{n-1}$	o_{n-2}					0.87	0.87	0.87	0.87
$a_{n-3}\ a_{n-2}\ a_{n-1}$	o_{n-3}					0.64	0.64	0.64	0.64
$a_1\ a_2\ a_3$	o_1	1.00	0.96	0.97	0.97	0.97	0.97	0.97	0.97
$a_1\ a_2\ a_3$	o_2	1.00	0.91	0.92	0.91	0.92	0.91	0.92	0.91
$a_2\ a_3\ a_4$	o_2		0.56	0.55	0.55	0.55	0.55	0.55	0.56
$a_1\ a_2\ a_3$	o_3	1.00	0.66	0.73	0.71	0.72	0.72	0.72	0.72
$a_2\ a_3\ a_4$	o_3		0.86	0.77	0.78	0.78	0.78	0.78	0.78
$a_3\ a_4\ a_5$	o_3			0.67	0.66	0.66	0.66	0.66	0.66
$a_2\ a_3\ a_4$	o_4		0.94	0.64	0.7	0.68	0.69	0.69	0.69
$a_3\ a_4\ a_5$	o_4			0.88	0.79	0.81	0.8	0.8	0.8
$a_4\ a_5\ a_6$	o_4				0.63	0.62	0.62	0.62	0.62
$a_3\ a_4\ a_5$	o_5			0.95	0.65	0.71	0.69	0.7	0.7
$a_4\ a_5\ a_6$	o_5				0.87	0.78	0.79	0.79	0.79
$a_5\ a_6\ a_7$	o_5					0.64	0.63	0.63	0.64
$a_4\ a_5\ a_6$	o_6				0.94	0.64	0.71	0.69	0.7
$a_5\ a_6\ a_7$	o_6					0.87	0.78	0.8	0.8
$a_6\ a_7\ a_8$	o_6						0.64	0.62	0.63
$a_5\ a_6\ a_7$	o_7					0.95	0.64	0.71	0.69
$a_6\ a_7\ a_8$	o_7						0.87	0.78	0.8
$a_6\ a_7\ a_8$	o_8						0.95	0.64	0.71
$a_1\ a_2\ a_3$	o_4		0.12	0.1	0.11	0.11	0.11	0.11	0.11
$a_2\ a_3\ a_4$	o_5			0.1	0.09	0.1	0.09	0.1	0.1
$a_3\ a_4\ a_5$	o_6				0.11	0.1	0.1	0.1	0.11
$a_4\ a_5\ a_6$	o_7					0.11	0.09	0.1	0.11
$a_5\ a_6\ a_7$	o_8						0.11	0.09	0.11

Table B.3: **Highest Relative Information Gain Contiguous Triples**, for DP problems of size $3 \leq n \leq 10$. We only show the (input, output) pairs where at least three problem sizes have RelativeIG>0, and at least one with RelativeIG>0.1. a_{n-1} refers to the last element of the sequence, regardless of its actual id in the sequence.

B.4 Theoretical Results: Derivations

B.4.1 Error accumulates with larger parallel applications of an estimated function (*width*)

Here we provide formal statements and derivations to Propositions 3.4.1 and 3.4.2 shown in the main paper. The mathematical framework used is a simplified representation of how multi-step reasoning works, showing two quintessential reasoning types: independent applications of the same step, or consecutive applications of the same step. We take an error estimation and accumulation perspective, since transformers are still being investigated from a theoretical standpoint.

Proposition B.4.1. *Let $f_n(\mathbf{x}) = h_n(g(\mathbf{x},1), g(\mathbf{x},2)), \dots, g(\mathbf{x},n))$. Let $\widehat{h}_n, \widehat{g}, \widehat{f}_n$ be estimators of h_n, g, f_n respectively. Assume $\mathbb{P}(h_n = \widehat{h}_n) = 1$ and $\mathbb{P}(h_n(X) = h_n(Y) \mid X \neq Y) < c_n$, where $c_n < c$ for some constant $c \ll 1$ (i.e. $\widehat{h}_n$ perfectly estimates h_n, and h_n is almost injective). If $\mathbb{P}(g \neq \widehat{g}) = \epsilon > 0$ and errors in $\widehat{g}$ are independent, $\liminf_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \geq 1 - c$.*

Moreover, if $c_n \leq \beta\alpha^n$ for some some $\alpha \in (0,1)$ and $\beta > 0$, then $\lim_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) = 1$.

Proof. For ease of writing, let $X_i = g(X,i)$ and $Y_i = \widehat{g}(X,i)$, and let $\boldsymbol{X} = (X_1, \dots, X_n)$ and $\boldsymbol{Y} = (Y_1, \dots, Y_n)$. We will compute some auxiliary probabilities, and then upper bound $\mathbb{P}(f = \widehat{f})$, to finally compute its limit.

$$\begin{aligned}\mathbb{P}(\boldsymbol{X} = \boldsymbol{Y}) &= \mathbb{P}(X_1 = Y_1, X_2 = Y_2, \dots, X_n = Y_n) \\ &= \mathbb{P}(X_1 = Y_1) \cdot \mathbb{P}(X_2 = Y_2) \dots \cdot \mathbb{P}(X_n = Y_n) = \mathbb{P}(g = \widehat{g})^n = (1-\epsilon)^n \end{aligned} \tag{B.1}$$

Since by hypothesis we know $\mathbb{P}(h_n(\boldsymbol{Y}) = \widehat{h}_n(\boldsymbol{Y})) = 1$, we have that:

$$\begin{aligned}\mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) &= \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y}) \cap h_n(\boldsymbol{Y}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \\ &= \mathbb{P}(h_n(\boldsymbol{X}) = h_n(\boldsymbol{Y}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \\ &\leq \mathbb{P}(h_n(\boldsymbol{X}) = h_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \\ &< c_n \end{aligned} \tag{B.2}$$

We will now estimate $\mathbb{P}(f_n = \widehat{f}_n)$ using the law of total probability w.r.t. the event $\boldsymbol{X} = \boldsymbol{Y}$.

$$\begin{aligned}\mathbb{P}(f_n = \widehat{f}_n) &= \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y})) \\ &= \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} = \boldsymbol{Y}) \cdot \mathbb{P}(\boldsymbol{X} = \boldsymbol{Y}) + \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \cdot \mathbb{P}(\boldsymbol{X} \neq \boldsymbol{Y}) \\ &= \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{X})) \cdot \mathbb{P}(\boldsymbol{X} = \boldsymbol{Y}) + \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \cdot (1 - \mathbb{P}(\boldsymbol{X} = \boldsymbol{Y})) \\ &= 1 \cdot (1-\epsilon)^n + \mathbb{P}(h_n(\boldsymbol{X}) = \widehat{h}_n(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \cdot (1 - (1-\epsilon)^n) \qquad \text{(using B.1 and hypothesis)} \\ &< (1-\epsilon)^n + c_n \cdot (1 - (1-\epsilon)^n) \qquad \text{(using B.2)} \\ &< c_n + (1-\epsilon)^n \cdot (1 - c_n)\end{aligned}$$

To conclude our proof, we will compute a lower bound for $\liminf_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n)$. Note that since $c_n < c$ for all n, we know that $\mathbb{P}(f_n = \widehat{f}_n) < c + (1-\epsilon)^n \cdot (1-c)$. Then,

$\mathbb{P}(f_n \neq \widehat{f}_n) > 1-c-(1-\epsilon)^n \cdot (1-c)$. Since $1-\epsilon \in [0,1)$, $\lim_{n\to+\infty} 1-c-(1-\epsilon)^n \cdot (1-c) = 1-c$. Thus,

$$\liminf_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \geq \liminf_{n\to+\infty} 1-c-(1-\epsilon)^n \cdot (1-c) = 1-c$$

which concludes our proof.

Note: In the case where $c_n \leq \beta\alpha^n$, we can derive an even stronger conclusion. In this case, we can prove that $\lim_{n\to+\infty} \mathbb{P}(f_n = \widehat{f}_n) = 1$. Recall that $\mathbb{P}(f_n = \widehat{f}_n) < \alpha\beta^n + (1-\epsilon)^n \cdot (1-\alpha\beta^n)$. Note that since $1-\epsilon \in [0,1)$ and $\alpha \in (0,1)$, trivially $\lim_{n\to+\infty} \beta\alpha^n + (1-\epsilon)^n \cdot (1-\beta\alpha^n) = 0$.

$$0 \leq \liminf_{n\to+\infty} \mathbb{P}(f_n = \widehat{f}_n) \leq \limsup_{n\to+\infty} \mathbb{P}(f_n = \widehat{f}_n) \leq \limsup_{n\to+\infty} \beta\alpha^n + (1-\epsilon)^n \cdot (1-\beta\alpha^n) = 0$$

Then, $\lim_{n\to+\infty} \mathbb{P}(f_n = \widehat{f}_n) = 0$ and we conclude $\lim_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) = 1$, assuming that $c_n \leq \beta\alpha^n$ for some adequate α, β. □

Corollary B.4.1. *Assume that a model $\mathcal{M}$ solves shifted addition perfectly, but it incorrectly solves **at least one** m digit by 1 digit multiplication for some fixed m. Then, the probability that $\mathcal{M}$ will solve **any** m digit by n digit multiplication using the long-form multiplication algorithm tends to 0 when n tends to infinity.*

Proof. Let $g = d \circ s$ define the base-10 multiplication between m-digit numbers $(x_1 x_2 \dots x_m)$ and 1-digit numbers (x_{m+j}), where $s : \mathbb{Z}_{10}^{m+n} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ denotes the selection of the numbers to multiply and $d : \mathbb{N} \times \mathbb{Z}_{10} \to \mathbb{N}$ denotes the actual multiplication:

$$\begin{aligned} g &:= d \circ s \\ d(x,y) &:= x \cdot y \\ s([x_1, \dots, x_m, x_{m+1}, \dots, x_{m+n}], j) &:= (x_1 +\!\!+\, x_2 +\!\!+\, \dots +\!\!+\, x_m,\ x_{m+j}) \\ &\qquad \text{where } x_1 +\!\!+\, x_2 +\!\!+\, \dots +\!\!+\, x_m \text{ denotes concatenating digits } x_i \end{aligned}$$

Let $h_n : \mathbb{N}^n \to \mathbb{N}$ describe the shifted addition used at the end of long-form multiplication to combine n m-digit by 1-digit multiplications, and let $f_n : \mathbb{Z}_{10}^{m+n} \to \mathbb{N}$ describe the long-form multiplication of m-digit by n-digit numbers:

$$\begin{aligned} h_n(x_1, \dots, x_n) &:= \sum_{i=1}^{n} x_i 10^{n-i} \\ f_n(\mathbf{x}) &:= h_n(g(\mathbf{x}, 1), g(\mathbf{x}, 2)), \dots, g(\mathbf{x}, n)) \end{aligned}$$

By hypothesis, $\mathbb{P}(g \neq \widehat{g}) = \epsilon > 0$ and $\mathbb{P}(h_n = \widehat{h}_n) = 1$, where $\widehat{g}$ and $\widehat{h}_n$ denote estimators using model $\mathcal{M}$. It can be shown that $\mathbb{P}(h_n(X) = h_n(Y) \mid X \neq Y) < \beta\alpha^n$ for $\alpha = 0.1$ and $\beta = 10^m$. Using Lemma B.4.1, $\lim_{n\to+\infty} \mathbb{P}(f_n \neq \widehat{f}_n) = 1$, which concludes our proof.

□

Note that Lemma B.4.1's proofs gives us empirical bounds once ϵ and α are approximated. Also **note that our definition of g in the proof of Corollary B.4.1 highlights two possible sources of exponentially-accumulating error**: errors in the selection of the numbers to multiply s, and errors in the actual m-digit by 1-digit multiplication d.

B.4.2 Error accumulates with larger iterative applications of an estimated function (*depth*)

Proposition B.4.2. *Let $f_n(\mathbf{x}) = g^n(\mathbf{x})$. Assume $\mathbb{P}(g(X) = \widehat{g}(Y) \mid X \neq Y) \leq c$ (i.e. recovering from a mistake due to the randomness of applying the estimator on an incorrect input has probability at most c). If $\mathbb{P}(g \neq \widehat{g}) = \epsilon > 0$ with $c + \epsilon < 1$, then $\liminf_{n \to +\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \geq 1 - c/(c+\epsilon)$.*

Proof. We first derive a recursive upper bound using the law of total probability, and then prove a non-recursive upper bound by induction.

$$\begin{aligned}
s_n := \mathbb{P}(f_n = \widehat{f}_n) &= \mathbb{P}(g(g^{n-1}(Z)) = \widehat{g}(\widehat{g}^{n-1}(Z))) \\
&= \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{Y})) \qquad \text{where } \boldsymbol{X} := g^{n-1}(Z) \text{ and } \boldsymbol{Y} := \widehat{g}^{n-1}(Z) \\
&= \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{Y}) \mid \boldsymbol{X} = \boldsymbol{Y}) \cdot \mathbb{P}(\boldsymbol{X} = \boldsymbol{Y}) + \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \cdot \mathbb{P}(\boldsymbol{X} \neq \boldsymbol{Y}) \\
&= \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{X})) \cdot \mathbb{P}(\boldsymbol{X} = \boldsymbol{Y}) + \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \cdot (1 - \mathbb{P}(\boldsymbol{X} = \boldsymbol{Y})) \\
&= \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{X})) \cdot s_{n-1} + \mathbb{P}(g(\boldsymbol{X}) = \widehat{g}(\boldsymbol{Y}) \mid \boldsymbol{X} \neq \boldsymbol{Y}) \cdot (1 - s_{n-1}) \\
&\leq (1 - \epsilon) \cdot s_{n-1} + c \cdot (1 - s_{n-1}) \\
&\leq (1 - \epsilon - c) \cdot s_{n-1} + c
\end{aligned}$$

We know $s_1 = (1 - \epsilon)$ since $s_1 = \mathbb{P}(f_1 = \widehat{f}_1) = \mathbb{P}(g = \widehat{g})$. Let $b := 1 - \epsilon - c$ for ease of writing. Then, we have

$$s_n \leq b \cdot s_{n-1} + c \tag{B.3}$$

It can be easily shown by induction that $s_n \leq b^{n-1}(1 - \epsilon) + c \sum_{i=0}^{n-2} b^i$:

- The **base case** $n = 2$ is true since we know $s_2 \leq b \cdot s_1 + c$, and $b \cdot s_1 + c = b(1-\epsilon) + c = b^{2-1}(1 - \epsilon) + c \sum_{i=0}^{2-2} b^i$, thus showing $s_2 \leq b^{2-1}(1 - \epsilon) + c \sum_{i=0}^{2-2} b^i$
- The **inductive step** yields directly using Equation B.3,

$$\begin{aligned}
&s_n \leq b \cdot s_{n-1} + c \\
&\quad \leq b \cdot \left(b^{n-2}(1 - \epsilon) + c \sum_{i=0}^{n-3} b^i \right) + c \leq b^{n-1}(1 - \epsilon) + c \sum_{i=1}^{n-2} b^i + c \leq b^{n-1}(1 - \epsilon) + c \sum_{i=0}^{n-2} b^i
\end{aligned}$$

We can rewrite the geometric series $\sum_{i=0}^{n-2} b^i$ in its closed form $\frac{1-b^{n-1}}{1-b}$, and recalling $b := 1 - \epsilon - c$,

$$\begin{aligned}
s_n &\leq b^{n-1}(1 - \epsilon) + c \frac{1 - b^{n-1}}{1 - b} = b^{n-1}(1 - \epsilon) + c \frac{1 - b^{n-1}}{c + \epsilon} \\
&= b^{n-1}(1 - \epsilon) + \frac{c}{c + \epsilon} - b^{n-1} \frac{c}{c + \epsilon} \\
&= b^{n-1} \left(1 - \epsilon - \frac{c}{c + \epsilon} \right) + \frac{c}{c + \epsilon}
\end{aligned}$$

Recalling that $s_n = \mathbb{P}(f_n = \widehat{f}_n)$, we compute the limit inferior of $\mathbb{P}(f_n \neq \widehat{f}_n) = 1 - s_n \geq 1 - b^{n-1}(1 - \epsilon - \frac{c}{c+\epsilon}) - \frac{c}{c+\epsilon}$.

$$\liminf_{n \to +\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \geq \lim_{n \to +\infty} 1 - b^{n-1} \left(1 - \epsilon - \frac{c}{c + \epsilon} \right) - \frac{c}{c + \epsilon} = 1 - \frac{c}{c + \epsilon}$$

that concludes our proof. □

We can generalize the proof in Lemma 3.4.2 to tasks where there are potentially many valid reasoning chains with the following alternative state-transition framing.

Lemma B.4.2. *Let S denote the set of all possible states a language model can generate, and let $z : S \to \{0, 1\}$ defines if a state is valid (0 = invalid). Let $\widehat{g} : S \to \Pi(S)$ be a state-transition function representing a language model's probability distribution of generating each possible next state when attempting to perform a single reasoning step. Assume $\mathbb{P}(z(\widehat{g}(X)) = 1 \mid z(X) = 0) \leq c$ and $\mathbb{P}(z(\widehat{g}(X)) = 0 \mid z(X) = 1) = \epsilon > 0$ with $c + \epsilon < 1$. Then, $\liminf_{n \to +\infty} \mathbb{P}(z(\widehat{g}^n) = 0) = 1 - c/(c + \epsilon)$.*

If for task T we know that all valid reasoning chains to arrive at a correct result have at least length n (i.e., the equivalent of defining $f_n = g^n$ in Lemma B.4.1) then the probability of solving task T correctly tends to at most $c/(c + \epsilon)$.

Corollary B.4.3. *The recursions for dynamic programming tasks and the m-by-1 digit multiplication are all tasks where there is a fixed reasoning step g being repeatedly applied. Therefore, we can directly apply Proposition 3.4.2 to these tasks.*

Proof. Let's analyze the two tasks separately below.

m-by-1 digit multiplication may be viewed as $f^m(\mathbf{x})$ Let $x = (x_1, \ldots, x_m)$ be the m-digit number that we multiply by the 1-digit number y ($0 \leq y < 10$). Let $z = (z_1, \ldots, z_{m+1})$ denote $z = x \cdot y$, which is guaranteed to have exactly $m + 1$ digits (with possibly leading zeros). We define f as:

$$f(x_1, \ldots, x_m, y, i, c) := (x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots x_m, y, i - 1, c')$$

where $x'_i := (x_i \cdot y + c) \mod 10$ and $c' := \lfloor (x_i \cdot y + c)/10 \rfloor$. Note that $x'_i = z_{i+1}$ since f is performing one step of the long-form multiplication algorithm.

Let the initial input be $\mathbf{x} := (x_1, \ldots, x_m, y, m, 0)$. Then, it can be easily shown that $f^m(\mathbf{x}) = (z_2, \ldots, z_{m+1}, y, 0, c)$. Since c is the left-most carry, it is the leading digit of z, i.e. $c = z_1$ (possibly zero) . Thus, the value of z can be directly extracted from $f^m(\mathbf{x}) = (z_2, \ldots, z_{m+1}, y, 0, z_1)$.

In the DP task, *dp*'s computation may be viewed as $f^{m-2}(x)$ for a list of size m See §B.1.2.1 for details on the solution to this problem. We will use identical notation. Let $a_1, \ldots, a_m$ be an input list. Let $\mathbf{x} = (a_1, \ldots, a_{m-2}, a'_{m-1}, a'_m, m - 2)$, where $a'_m := \max(a_m, 0)$ and $a'_{m-1} := \max(a_{m-1}, a_m, 0)$. Intuitively, this means that we have applied the first two steps of the dp computation, and stored the results in a'_{m-1} and a'_m. Let f be a function representing the recursive computation of dp_i:

$$f(a_1, \ldots, a_i, a'_{i+1}, \ldots, a'_m, i) = (a_1, \ldots, a_{i-1}, a'_i, \ldots, a'_m, i - 1)$$

where $a'_i := \max(a'_{i+1}, a_i + a'_{i+2}, 0)$.

Note that since a'_{i+1} stores the value of dp_{i+1} and a'_{i+2} stores the value of dp_{i+2}, it can be easily shown that $f^{m-2}(\mathbf{x}) = (a'_1, \ldots, a'_m, 0) = (dp_1, \ldots, dp_m, 0)$. Therefore, f^{m-2} computes all recursive values of dp_i when given the base cases.

In the DP task, the reconstruction of the desired subsequence given already computed dp values may be viewed as $f^m(x)$ for an input list of size m. This case is similar to the previous one. Let $r = (r_1, \ldots, r_m)$ be the result, where $r_i = 1$ if a_i was selected for the desired subsequence, and $r_i = 2$ otherwise. Let $\mathbf{x} := (dp_1, \ldots, dp_m, 0, 0, a_1, \ldots, a_m, 1, 1)$. Let f be defined as follows:

$$f(dp_1, \ldots, dp_m, 0, 0, a'_1, \ldots, a'_{i-1}, a_i, \ldots, a_m, i, u) = (dp_1, \ldots, dp_m, 0, 0, a'_1, \ldots, a'_i, a_{i+1}, \ldots, a_m, i+1, u')$$

where $a'_i := 2 - \mathbb{1}\{dp_i = a_i + dp_{i+2} \text{ and } u = 1\}$ and $u := 1 - \mathbb{1}\{dp_i = a_i + dp_{i+2} \text{ and } u = 1\}$. Intuitively, a'_i stores whether the i-th element of the list should be selected for the final subsequence, assigning 1 if the element should be taken, and 2 otherwise (i.e., $a'_i = r_i$). Moreover, if the i-th element has been selected, we mark that the next item will not be available using u'. Therefore, f performs one step of the final output reconstruction as defined in §B.1.2.1.

It can be easily shown that $f^m(\mathbf{x}) := (dp_1, \ldots, dp_m, 0, 0, a'_1, \ldots, a'_m, m+1, u') = (dp_1, \ldots, dp_m, 0, 0, r_1, \ldots, r_m, m+1, u')$. Note that the extra two elements in the input state allow lifting the special cases $m-1$ and m in the solution shown in §B.1.2.1 without falling out of bounds.

□

B.4.3 Discussing $c \ll \epsilon$ in the context of Proposition 4.2

Note that in Proposition 3.4.2, if $c \ll \epsilon$ then $\liminf_{n \to +\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \approx 1$. This is because assuming $\epsilon = m \cdot c$ for some $m > 0$, we have $1 - \frac{c}{c+\epsilon} = 1 - \frac{c}{c+m \cdot c} = 1 - \frac{1}{m+1} = \frac{m}{m+1}$, and $\frac{m}{m+1}$ is a monotonically increasing function for all $m > 0$ that tends to 1 when m goes to infinity. Therefore, large m's (or alternatively, $c \ll \epsilon$) imply $\frac{m}{m+1}$ will be close to 1.

It is reasonable to assume $c \ll \epsilon$ when g has low collision, since c represents the probability of the estimator $\widehat{g}(y)$ arriving at the correct output $g(x)$ by chance when given the wrong input $y \neq x$.

If g is discrete, it can take $|\mathrm{Im}(g)|$ values, where $|\mathrm{Im}(g)|$ denotes the cardinal of the image space of g. Assuming approximately uniform errors, $c \approx \epsilon / |\mathrm{Im}(g)|$, which in turn implies $c \ll \epsilon$ since g being low collision implies $|\mathrm{Im}(g)|$ is large.

If g is continuous, under appropriate assumptions it seems plausible that we can prove that $c \approx 0$ (e.g. if errors are approximately uniform).

Summarizing both cases, if errors are approximately evenly distributed we obtain that $\liminf_{n \to +\infty} \mathbb{P}(f_n \neq \widehat{f}_n) \approx 1$.

Appendix C

SymbolicToM: Additional Details and Experiments

C.1 Additional Details on SymbolicToM

C.1.1 Detailed Description of Information Contained in Global Context G

In the main paper, we define G as a graph containing the true world state (as opposed to beliefs about the current world state). This means that G will represent where people and objects are truly located, regardless of beliefs. G will in general contain only the *observable* true world state. Thus, information passed verbally would not be stored in the global context (e.g. someone speaking in a room is not observable after they finished talking), and would instead be stored in the local contexts of the people that heard the speech. Since verbal interactions are not tested by available datasets, this distinction is not relevant in ToMi.

C.1.2 Prompts for Resulting State Extraction

For GPT3-Curie we 2-shot prompt with the following prompt (both for original and linguistic diversity experiments):

```
John quit his job.  The resulting state after this action is that John no
longer has a job.\n\nJohn signed a contract.  The resulting state after this
action is that the contract is signed.\n\n<sentence>.  The resulting state
after this action is that
```

We find that GPT3-Davinci, Flan-T5-XL, GPT3.5, and GPT4 are able to zero-shot answer to this subtask just by describing the instruction, but smaller models benefit from few-shot. We were unable to query Macaw for this task, so we instead rely on GPT3-Curie, a model of comparable size. Zero-shot instruction is as follows:

```
<sentence>.  What is the resulting state after this action?  Do not add
new information.  The resulting state after this action is that
```

We observe that GPT3 is significantly more robust to paraphrases than Flan-T5: Flan-T5

models are poor at detecting the resulting state for florid paraphrases, although the original phrasings are a straightforward task for Flan-T5.

Larger models like GPT3.5 and GPT4 are able to perform the task well zero-shot, similarly to GPT3; LLaMA models require fewer demonstrations than Flan-T5. We ran all main experiments implementing Resulting State Extraction with GPT3.

C.1.3 Solving processQuestion using GPT3

Our explorations suggest that GPT3 (Curie and GPT3-Davinci `text-davinci-002`—the version used in all our experiments) can successfully extract entities and rephrase the question. See Figure C.1 for an example prompt.

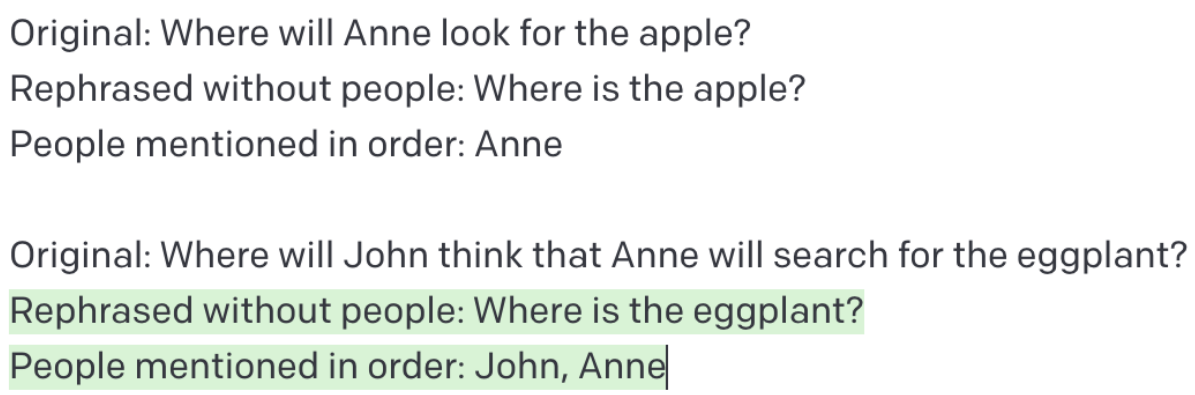


Figure C.1: GPT3 shows one-shot generalization abilities from first-order to second-order questions.

C.2 Details on Out-Of-Domain Evaluation

C.2.1 Linguistic Diversity Per ToMi Template

C.2.1.1 All Paraphrases of `PersonX entered the RoomY.`

```
PersonX entered the RoomY.
PersonX approached the RoomY.
PersonX arrived at the RoomY.
PersonX arrived in the RoomY.
PersonX bounded into the RoomY.
PersonX came by the RoomY.
PersonX came into the RoomY.
PersonX came to the RoomY.
PersonX crept into the RoomY.
PersonX entered the RoomY.
PersonX leapt into the RoomY.
```

Sentence type	*Count*
Object's Position	38
Distractor Negative Sentiment	36
Distractor Positive Sentiment	31
Person Entered Room	21
Person Exited Room	19
Person Moved Object	18
Person's Position	9

Table C.1: Number of paraphrases per original sentence template. Paraphrases were obtained from prompting GPT3-Davinci (`text-davinci-002`).

```
PersonX showed up at the RoomY.
PersonX shuffled into the RoomY.
PersonX sidled into the RoomY.
PersonX slithered into the RoomY.
PersonX stepped into the RoomY.
PersonX tiptoed into the RoomY.
PersonX visited the RoomY.
PersonX walked into the RoomY.
PersonX went into the RoomY.
PersonX went to the RoomY.
```

C.2.1.2 All Paraphrases of `PersonX exited the RoomY.`

Prompted with the prompt: *Find 30 alternative ways of expressing the following sentence: Abigail exited the bedroom.* and manually filtering results (with this and other name/location selection.

```
PersonX exited the RoomY.
PersonX left the RoomY.
PersonX walked out of the RoomY.
PersonX stepped out of the RoomY.
PersonX departed the RoomY.
PersonX went out of the RoomY.
PersonX came out of the RoomY.
PersonX emerged from the RoomY.
PersonX quit the RoomY.
PersonX took off from the RoomY.
PersonX bolted from the RoomY.
PersonX flew from the RoomY.
PersonX ran from the RoomY.
PersonX sprinted from the RoomY.
PersonX jogged from the RoomY.
PersonX hurried from the RoomY.
```

PersonX crawled from the RoomY.
PersonX crept from the RoomY.
PersonX tiptoed from the RoomY.

C.2.1.3 All Paraphrases of `The Object1 is in the Container1.`

Prompted with `Object1=apple`, `Container1={fridge, envelope, bathtub}`. Then filtered to remove object-specific wording.

The Object1 is in the Container1.
The Object1 is stored in the Container1.
The Object1 is kept in the Container1.
The Object1 is located in the Container1.
The Object1 is situated in the Container1.
The Object1 is set in the Container1.
The Object1 is placed in the Container1.
The Object1 is found in the Container1.
The Object1 is positioned in the Container1.
The Object1 is set upon in the Container1.
The Object1 is put in the Container1.
The Object1 is laid in the Container1.
The Object1 is deposited in the Container1.
The Object1 is stationed in the Container1.
The Object1 is put to rest in the Container1.
The Object1 is set to rest in the Container1.
The Object1 is rested in the Container1.
The Object1 is set aside in the Container1.
The Object1 is stowed in the Container1.
The Container1 contains the Object1.
The Object1 is inside the Container1.
The Object1 is within the Container1.
The Container1 is where the Object1 is.
The Container1 has the Object1.
The Container1 is holding the Object1.
The Container1 is keeping the Object1.
The Container1 is safeguarding the Object1.
The Container1 is storing the Object1.
The Container1 has the Object1 within it.
The Container1 has the Object1 inside of it.
The Container1 is holding the Object1 within it.
The Container1 is keeping the Object1 inside of it.
The Container1 is safeguarding the Object1 inside of it.
The Container1 is storing the Object1 inside of it.
There is a Object1 in the Container1.
A Object1 is in the Container1.

The Container1 has a Object1 in it.
Inside the Container1 is a Object1.

C.2.1.4 All Paraphrases of `PersonX moved the Object1 to the Container1.`

PersonX moved the Object1 to the Container1.
PersonX relocated the Object1 to the Container1.
PersonX transferred the Object1 to the Container1.
PersonX shifted the Object1 to the Container1.
PersonX placed the Object1 in the Container1.
PersonX set the Object1 in the Container1.
PersonX put the Object1 in the Container1.
PersonX stowed the Object1 in the Container1.
PersonX stored the Object1 in the Container1.
PersonX hid the Object1 in the Container1.
PersonX shoved the Object1 into the Container1.
PersonX pushed the Object1 to the Container1.
PersonX carried the Object1 to the Container1.
PersonX conveyed the Object1 to the Container1.
PersonX led the Object1 to the Container1.
PersonX transported the Object1 to the Container1.
PersonX brought the Object1 to the Container1.
PersonX took the Object1 to the Container1.

C.2.1.5 All Paraphrases of `PersonX is in the RoomY.`

PersonX is in the RoomY.
PersonX is inside the RoomY.
PersonX is located in the RoomY.
PersonX is situated in the RoomY.
PersonX is present in the RoomY.
PersonX is to be found in the RoomY.
PersonX is contained in the RoomY.
The RoomY holds PersonX.
The RoomY shelters PersonX.

C.2.1.6 All Paraphrases of Positive Distractor Sentences

PersonX has a bad case of Object1 fever.
PersonX is Object1 crazy.
PersonX is Object1-crazed.
PersonX is Object1-obsessed.
PersonX is a Object1 fiend.
PersonX is a Object1 maniac.
PersonX is a Object1-aholic.
PersonX is always thirsty for a Object1.

PersonX is besotted with the Object1.
PersonX is captivated by the Object1.
PersonX is charmed by the Object1.
PersonX is crazy about the Object1.
PersonX is crazy for the Object1.
PersonX is eager for the Object1.
PersonX is enamored with the Object1.
PersonX is enthusiastic about the Object1.
PersonX is entranced by the Object1.
PersonX is fascinated by the Object1.
PersonX is fond of the Object1.
PersonX is in love with the Object1.
PersonX is infatuated with the Object1.
PersonX is keen on the Object1.
PersonX is mad about the Object1.
PersonX is never seen without a Object1.
PersonX is nuts about the Object1.
PersonX is smitten with the Object1.
PersonX is spellbound by the Object1.
PersonX is taken with the Object1.
PersonX is wild about the Object1.
PersonX loves to drink from a Object1.
PersonX would do anything for a Object1.

C.2.1.7 All Paraphrases of Negative Distractor Sentences (`PersonX hates ObjectY`)

PersonX hates Object1.
PersonX can't stand the Object1.
PersonX despises the Object1.
PersonX detests the Object1.
PersonX is annoyed by the Object1.
PersonX is bothered by the Object1.
PersonX is concerned by the Object1.
PersonX is disconcerted by the Object1.
PersonX is discouraged by the Object1.
PersonX is disgusted by the Object1.
PersonX is disheartened by the Object1.
PersonX is disquieted by the Object1.
PersonX is grieved by the Object1.
PersonX is horrified by the Object1.
PersonX is irritated by the Object1.
PersonX is offended by the Object1.
PersonX is pained by the Object1.
PersonX is repelled by the Object1.

```
PersonX is revolted by the Object1.
PersonX is scandalized by the Object1.
PersonX is shocked by the Object1.
PersonX is sorrowful by the Object1.
PersonX is terrified by the Object1.
PersonX is troubled by the Object1.
PersonX is vexed by the Object1.
PersonX loathes the Object1.
The Object1 horrifies PersonX.
The Object1 is abhorrent to PersonX.
The Object1 nauseates PersonX.
The Object1 offends PersonX.
The Object1 repulses PersonX.
The Object1 revolts PersonX.
The Object1 scandalizes PersonX.
The Object1 shocks PersonX.
The Object1 sickens PersonX.
The Object1 terrifies PersonX.
The Object1 turns PersonX's stomach.
```

C.2.2 Structure of Story Structure Robustness Test Sets

C.2.2.1 Double Room False-Belief Episode

```
person1 entered the room1.
person2 entered the room1.
The object1 is in the container1.
The container1 is in the room1.
person2 exited the room1.
person1 moved the object1 to the container2.
The container2 is in the room1.
person1 exited the room1.
person2 entered the room2.
person1 entered the room2.
The object2 is in the container3.
The container3 is in the room2.
person1 exited the room2.
person2 moved the object2 to the container4.
The container4 is in the room2.
person2 exited the room2.
```

C.2.2.2 Three Active Characters Story

```
person1 entered the room1.
person2 entered the room1.
```

```
person3 entered the room1.
The object1 is in the container1.
The container1 is in the room1.
person2 exited the room1.
person1 moved the object1 to the container2.
The container2 is in the room1.
person1 exited the room1.
person3 moved the object1 to the container3.
The container3 is in the room1.
person3 exited the room1.
```

C.2.2.3 Four Containers with Multiple Movements

```
person1 is in the room1.
The object1 is in the container1.
The container1 is in the room1.
person1 moved the object1 to the container2.
The container2 is in the room1.
person2 entered the room1.
person1 exited the room1.
person2 moved the object1 to the container3.
The container3 is in the room1.
person2 moved the object1 to the container4.
The container4 is in the room1.
```

C.3 Expanded Results

Experimental Note: All zero-shot GPT3 (`text-curie-001` and `text-davinci-002`) experiments were performed between November 2022 and January 2023. GPT3.5 (`gpt-3.5-turbo`) and GPT4 (`gpt-4`) were added in May 2023.

C.3.1 Ablating filterBasedOnQuestion from SymbolicToM

filterBasedOnQuestion definition This function filters the story S' to obtain an even shorter subset of the original story S'' by only keeping edges where at least one of the endpoints represents an entity mentioned in the question.

Last step of Algorithm 2 is applying filterBasedOnQuestion, which yields an even shorter story to feed language models. We evaluate the effect this final filter has on the final performances reported by SymbolicToM.

filterBasedOnQuestion has a positive effect on Macaw-3B, GPT3, Flan-T5-XXL, and LLaMA-7B (+7, +3.5, +12.8, and +15 points in average accuracy gain across all question types), and a mild negative one on Flan-T5-XL, and GPT4 (-5.3, and -4 points of accuracy on average). See Table C.5 for all differences between executing SymbolicToM using this final

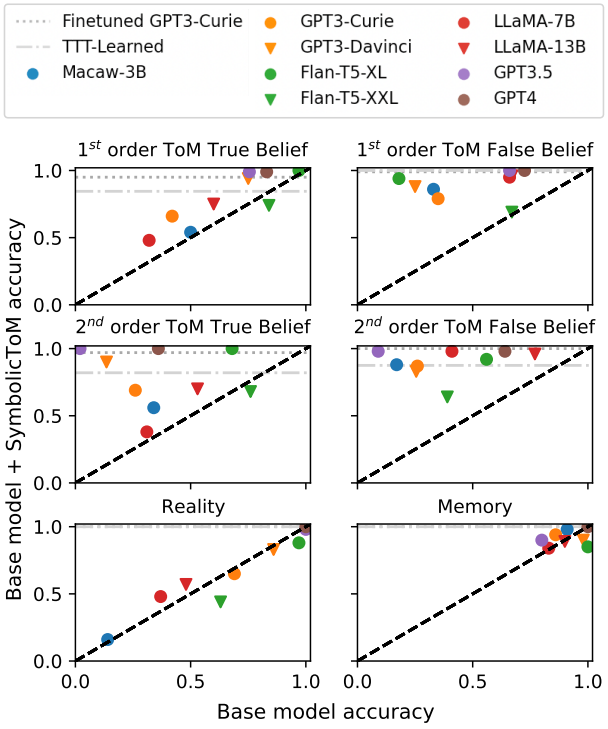


Figure C.2: Precision using SymbolicToM on ToMi, for several language models without the final filter function. Performance is shown for each question type, dots in upper triangle imply performance improvements. Full results table may be found in Table C.4.

filtering or not. Figure C.2 visually represents the accuracy of all models by question type. Regardless of the final filter application, GPT4+SymbolicToM significantly outperforms out-of-the-box GPT4 in all four ToM question types and maintains performance on Reality and Memory questions. For Flan-T5-XL, Flan-T5-XL+SymbolicToM outperforms Flan-T5-XL significantly in all four ToM question types (e.g. +76 and +36 points in accuracy for first and second-order false belief questions), and shows slight declines for Reality and Memory questions—in line with findings on the full algorithm, but with less stark declines, suggesting that having more entities may help reduce bias towards answering rooms instead of containers. See Table C.4 for the full table of accuracy differences.

Regardless of the final filtering application, SymbolicToM shows improvements in theory of mind questions for all models. We only find the filter application to be relevant to beat the base model in theory of mind questions for Flan-T5-XXL.

C.3.2 Third-Order Theory of Mind Evaluation

We ask all third-order theory of mind questions for each D_2 story, such as "Where does p_1 think that p_2 thinks that p_1 will search for the o_1?". Questions involving p_2 will have a final answer c_1, since everyone saw p_2 leaving. We ask all six possible questions involving p_2. We also ask the two third-order theory of mind questions that do not involve p_2 nor repeats the same person twice consecutively ("Where does p_1 think that p_3 thinks that p_1 will search for

the o_1?" and "Where does p_3 think that p_1 thinks that p_3 will search for the o_1?"), totaling eight questions per D_2 story.

Table C.2 shows results for all models using $k = 2$ representations (same depth as in the main paper). Using SymbolicToM significantly outperforms the supervised baselines and yields dramatic improvements with respect to using the LLMs off-the-shelf. We hypothesize that although the task theoretically requires $k = 3$, the second-order theory of mind representation already helps models avoid attending to parts of the story that are inaccessible to relevant characters.

C.3.3 Alternative resultingState Implementations

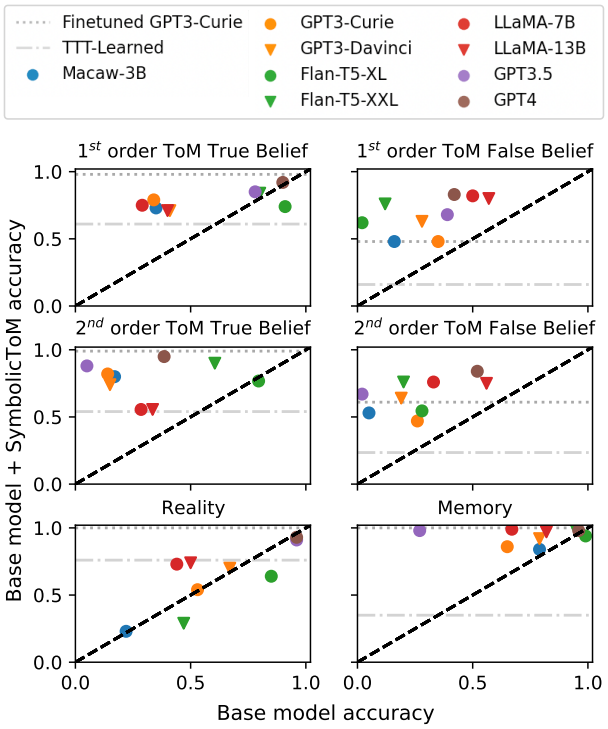


Figure C.3: Results for ParaphrasedToMi when using the same model for implementing the resultingState function as in the final question-answering task (except using Davinci for Macaw, who did not show reliable enough few shot-prompting). Dots in upper triangle imply performance with SymbolicToM is higher than using the base model out-of-the-box. Horizontal lines reflect supervised models' performance (higher is better).

resultingState(s) refers to the state of the world after s has been performed. For example, if "Oliver moved the apple to the box", then the resulting state is that "The apple is in the box". If "Oliver exited the bedroom", the resulting state would be that "Oliver is no longer in the bedroom". These are the relationships that we may insert in a context graph—actions are instantaneous and do not reflect an observable state.

In this section, we explore using the same LLM for implementing RESULTINGSTATE as well as the final inference. In the main text, we use Davinci for all non-GPT3-based models.

We find GPT3 to be among the most reliable to answer the resulting state of a given action in a zero-shot (Davinci) or two-shot (Curie) manner. Similarly, GPT3.5 and GPT4 perform well zero-shot: for experiments, we use GPT3.5 zero-shot and GPT4 two-shot to improve the resulting phrasing stability.

Additional exploration shows that although Flan-T5 models perform worse zero-shot than GPT models, they are capable of performing this task with more careful prompting. Figure C.3 shows the results after nine-shot prompting Flan-T5-XL and eleven-shot prompting Flan-T5-XXL. Our explorations show that LLaMA models require fewer demonstrations than the Flan-T5 models to compute the resulting state: we observe highly reliable results when using six-shot prompting for LLaMA-7B, and seven-shot prompting for LLaMA-13B. Accuracy using LLaMA was even higher than when using GPT3.

C.3.4 Detailed Result Tables

All results in the appendix show accuracy as a ratio (between 0 and 1). For simplicity of reading, in the main text, they are referred to in percentages (values 0 to 100, higher is better). Figures C.3, C.4, and C.5 show performances when applying the final filtering function, when not applying it, and the difference in performance between the two, respectively.

	D_2's third-order ToM questions
Off-the-shelf models	
Macaw-3B	13
Flan-T5-XL	32
Flan-T5-XXL	62
GPT3-Curie	28
GPT3-Davinci	19
GPT3.5	8
GPT4	26
LLaMA-7B	22
LLaMA-13B	39
SymbolicToM + Off-the-shelf models	
Macaw-3B	85 (+72)
Flan-T5-XL	97 (+65)
Flan-T5-XXL	**100** (+38)
GPT3-Curie	89 (+61)
GPT3-Davinci	90 (+71)
GPT3.5	**100** **(+91)**
GPT4	**100** (+73)
LLaMA-7B	90 (+68)
LLaMA-13B	95 (+57)
Supervised models	
TTT	52
Finetuned GPT3	76

Table C.2: Precision using SymbolicToM on all questions from 100 stories for each of the modified test sets D_i. Supervised models were trained on ToMi; all others do not require training. Parenthesis reflect differences between using and not using SymbolicToM: **bold** reflects higher overall performance, and **green** reflects the highest net improvements when using SymbolicToM.

	1st TB	1st FB	2nd TB	2nd FB	Reality	Memory
Macaw-3B	0.86 [0.50]	0.79 [0.33]	0.86 [0.34]	0.84 [0.17]	0.10 [0.14]	0.95 [0.91]
GPT3-Curie	0.77 [0.42]	0.82 [0.35]	0.73 [0.26]	0.89 [0.26]	0.61 [0.69]	0.99 [0.86]
GPT3-Davinci	0.96 [0.75]	0.96 [0.25]	0.93 [0.14]	0.90 [0.26]	0.77 [0.86]	0.98 [0.98]
Flan-T5-XL	0.98 [0.97]	0.80 [0.18]	0.98 [0.68]	0.78 [0.56]	0.73 [0.97]	1.00 [1.00]
Flan-T5-XXL	0.98 [0.84]	0.95 [0.67]	1.00 [0.76]	0.90 [0.39]	0.13 [0.63]	1.00 [1.00]
LLaMA-7B	0.82 [0.32]	0.95 [0.66]	0.66 [0.31]	0.72 [0.41]	0.87 [0.37]	1.00 [0.83]
LLaMA-13B	0.82 [0.60]	0.86 [0.67]	0.70 [0.53]	0.62 [0.77]	0.87 [0.48]	1.00 [0.90]
GPT3.5	0.97 [0.76]	0.95 [0.66]	0.99 [0.02]	0.87 [0.09]	0.98 [1.00]	0.99 [0.80]
GPT4	0.98 [0.83]	0.94 [0.73]	0.98 [0.36]	0.89 [0.64]	0.94 [1.00]	1.00 [1.00]
Finetuned GPT3	0.95	0.99	0.97	1.00	1.00	1.00
TTT-learned	0.84	1.00	0.82	0.88	1.00	1.00

Table C.3: Performance per model and question using SymbolicToM, with out-of-the-box performance shown in brackets (100 samples per question type). Bottom rows represent supervised baselines.

	1st TB	1st FB	2nd TB	2nd FB	Reality	Memory
Macaw-3B	0.54 [0.50]	0.86 [0.33]	0.56 [0.34]	0.88 [0.17]	0.16 [0.14]	0.98 [0.91]
GPT3-Curie	0.66 [0.42]	0.79 [0.35]	0.69 [0.26]	0.87 [0.26]	0.65 [0.69]	0.94 [0.86]
GPT3-Davinci	0.94 [0.75]	0.88 [0.25]	0.90 [0.14]	0.83 [0.26]	0.83 [0.86]	0.90 [0.98]
Flan-T5-XL	1.00 [0.97]	0.94 [0.18]	1.00 [0.68]	0.92 [0.56]	0.88 [0.97]	0.85 [1.00]
Flan-T5-XXL	0.74 [0.84]	0.69 [0.67]	0.68 [0.76]	0.64 [0.39]	0.44 [0.63]	1.00 [1.00]
LLaMA-7B	0.48 [0.32]	0.95 [0.66]	0.38 [0.31]	0.98 [0.41]	0.48 [0.37]	0.84 [0.83]
LLaMA-13B	0.75 [0.60]	0.96 [0.67]	0.70 [0.53]	0.96 [0.77]	0.57 [0.48]	0.89 [0.90]
GPT3.5	0.99 [0.76]	1.00 [0.66]	1.00 [0.02]	0.98 [0.09]	0.98 [1.00]	0.90 [0.80]
GPT4	0.99 [0.83]	1.00 [0.73]	1.00 [0.36]	0.98 [0.64]	1.00 [1.00]	1.00 [1.00]
Finetuned GPT3	0.95	0.99	0.97	1.00	1.00	1.00
TTT-learned	0.84	1.00	0.82	0.88	1.00	1.00

Table C.4: Performance per model and question using SymbolicToM without filterBasedOnQuestion, with out-of-the-box performance shown in brackets (100 samples per question type). Bottom rows represent supervised baselines.

	1st TB	1st FB	2nd TB	2nd FB	Reality	Memory
Macaw-3B	0.32	-0.07	0.30	-0.04	-0.06	-0.03
GPT3-Curie	0.11	0.03	0.04	0.02	-0.04	0.05
GPT3-Davinci	0.02	0.08	0.03	0.07	-0.06	0.08
Flan-T5-XL	-0.02	-0.14	-0.02	-0.14	-0.15	0.15
Flan-T5-XXL	0.24	0.26	0.32	0.26	-0.31	0.00
LLaMA-7B	0.34	0.00	0.28	-0.26	0.39	0.16
LLaMA-13B	0.07	-0.10	0.00	-0.34	0.30	0.11
GPT3.5	-0.02	-0.05	-0.01	-0.11	0.00	0.09
GPT4	-0.01	-0.06	-0.02	-0.09	-0.06	0.00

Table C.5: Differences between accuracy of base models using SymbolicToM with the final filterBasedOnQuestion filter, and without using the final filter. As shown in Table C.3 and C.4, both versions are still far superior to not using SymbolicToM.

Appendix D

ExploreToM: Additional Details and Experiments

D.1 Actions' formal definition (cont. from 5.2.2.1)

All actions are functions that transform a state into another state, updating the world state and the beliefs of everyone involved up to two levels of recursion. All actions have preconditions, e.g. to enter a room you need to not be in it already.

A state $\in \mathcal{S}$ is comprised of a world state ws (the things currently true physically about the world described), the first-order beliefs b_1, and the second-order beliefs b_2. First-order beliefs describe what each person believes to be the current world state, e.g. Anne believes that the apple is salted. Second-order beliefs describe what each person estimates that each other person believes to be the current world state, e.g. Anne believes that Beth thinks that the apple is salted.

Let's describe the definition of leaving in a room through an example: "Beth left the kitchen.", and build the definition of the action function $a_{\text{leave, Beth, kitchen}} : \mathcal{S} \to \mathcal{S}$. As described above, the state is comprised of a world state, first-order beliefs, and second-order beliefs, i.e.,

$$a_{\text{leave, Beth, kitchen}}(ws, b_1, b_2) := (ws', b_1', b_2')$$

Let's first describe the world state update ws'. The world state remains the same for every entity (object, container, person, etc.), except for the person leaving the room—Beth. Thus,

$$ws(q, \text{room}) \;=\; ws'(q, \text{room}) \;\; \forall q \neq \text{Beth} \quad \text{and} \quad ws'(q, \text{room}) = \neg\text{kitchen}$$

Let's then describe the first-order belief updates b_1'. Here, we assume that everyone in the same room as Beth (the kitchen) will know that Beth has left. We denote this group of people as witnesses(kitchen):

$$\text{witnesses(kitchen)} := \{p | ws(p, \text{room}) = \text{kitchen}\}$$

Everyone not in the kitchen will assume that Beth is still there unless communicated otherwise, since they have no reason to believe she has left. Thus,

$$b_1(p, \text{Beth}, \text{room}) = b'_1(p, \text{Beth}, \text{room}) = \text{kitchen} \quad \forall p \notin \text{witnesses}(\text{kitchen})$$

$$b_1(p, \text{Beth}, \text{room}) = \neg\text{kitchen} \quad \forall p \in \text{witnesses}(\text{kitchen})$$

We now describe the second-order belief updates b'_2. Here, we assume that everyone in the kitchen (including Beth) assumes that everyone else in the kitchen knows Beth left (and only them). If someone was not in the kitchen, they will assume nothing has happened. Formally,

$$b_2(p, q, \text{Beth}, \text{room}) = b'_2(p, q, \text{Beth}, \text{room}) = \text{kitchen} \quad \forall p \notin \text{witnesses}(\text{kitchen}), \quad \forall q$$

$$b_2(p, q, \text{Beth}, \text{room}) = \neg\text{kitchen} \quad \forall p \in \text{witnesses}(\text{kitchen}) \quad \forall q \in \text{witnesses}(\text{kitchen})$$

$$b_2(p, q, \text{Beth}, \text{room}) = \text{kitchen} \quad \forall p \in \text{witnesses}(\text{kitchen}) \quad \forall q \notin \text{witnesses}(\text{kitchen})$$

Finally, the function can only be applied if Beth is in the kitchen, i.e. it has the precondition $ws(\text{Beth}, \text{room}) = \text{kitchen}$.

All other functions definitions can be found verbatim in the code to be released.

D.2 All supported questions (Cont. from § 5.2.2.2)

Table D.1: List of all supported ExploreToM questions per property discussed and level of theory of mind, transcribed verbatim.

Property asked about	ToM Order	Question (requesting *Short Answer.* in prompt)	Expected Answers
room location	–	In which room was the <object> at the beginning?	room name
room location	–	In which room is the <object> now?	room name
room location	–	In which room was the <object> before <action>?	room name
room location	1st	In which room will <person> search for the <object>?	room name
room location	2nd	In which room does <person1> think that <person2> will search for the <object>?	room name
container location	–	In which container was the <object> at the beginning?	container name
container location	–	In which container is the <object> now?	container name
container location	–	In which container was the <object> before <action>?	container name
container location	1st	In which container will <person> search for the <object>?	container name
container location	2nd	In which container does <person1> think that <person2> will search for the <object>?	container name
abstract topic knowledge	1st	Does <person1> know about <topicDiscussed>?	*yes* or *no*
abstract topic knowledge	2nd	What does <person1> think about <person2>'s belief on <topicDiscussed>? (knows about it / does not know about it)	*knows about it* or *does not know about it*
knowledge about state update	1st	Does <person> believe that the <object> <newState>? Answer yes or no.	*yes* or *no*
knowledge about state update	2nd	Does <person1> believe that <person2> believes that the <object> <newState>? Answer yes or no.	*yes* or *no*

D.3 Additional Experiments

D.3.1 A*-generated stories are more challenging than overgenerating and filtering (cont. from § 5.3)

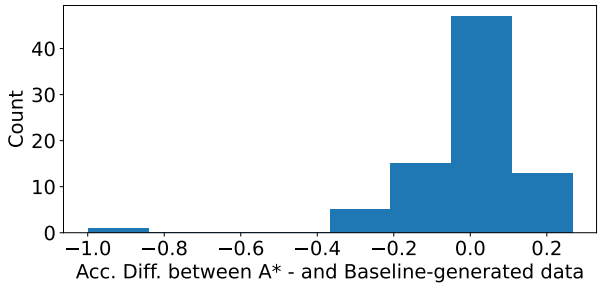


Figure D.1: Histogram depicting accuracy differences between A*-generated datasets for Llama-3.1-70B-Instruct and a dataset created by over-generating and filtering with the same budget (i.e., baseline). Results show that A* is better at finding story structures that make a challenging benchmark by showing low accuracy (negative values mean A* is better at finding challenging story structures).

D.3.2 Infilled Story Structures Remain Challenging (cont. from § 5.3)

Table D.2 shows a breakdown across all settings. 40% of the stories were infilled in a single attempt. Next step options are sampled simultaneously with repetition penalty for added

Table D.2: Changes in accuracy when infilling ExploreToM-generated story structures to output natural-sounding stories. We only include comparison between the 40% of stories where the LLM as a judge (Llama-3.1-70B Instruct) determined that all infilled actions were high quality.

Action Set: $\{a_{\text{enter}}, a_{\text{leave}}, \ldots$	Include asymmetry	Acc. Story Structure	Acc. Infilled	Acc. Diff.
$\ldots, a_{\text{moveObjContainer}}\}$, (denoted $\mathcal{A}_1$)	✗	0.20	0.43	0.22
	✓	0.02	0.36	0.35
$\ldots, a_{\text{updateObjState}}\}$, (denoted $\mathcal{A}_2$)	✗	0.40	0.44	0.03
	✓	0.49	0.48	-0.01
$\ldots, a_{\text{moveObjContainer}}, a_{\text{updateObjState}}\}$ ($\mathcal{A}_3$)	✗	0.45	0.48	0.03
	✓	0.17	0.60	0.43
$\ldots, a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}\}$ ($\mathcal{A}_4$)	✗	0.12	0.73	0.62
	✓	0.13	0.37	0.23
$\ldots, a_{\text{moveObjContainer}}, a_{\text{info-*}}\}$ ($\mathcal{A}_5$)	✗	0.09	0.45	0.37
	✓	0.10	0.42	0.32
$\ldots, a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}, a_{\text{info-*}}\}$ ($\mathcal{A}_6$)	✗	0.08	0.31	0.23
	✓	0.13	0.51	0.38
$a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}, a_{\text{chitChat-*}}, a_{\text{info-*}}\}$ ($\mathcal{A}_7$)	✗	0.73	0.86	0.13
	✓	0.75	0.85	0.10
$\ldots, a_{\text{chitChat-private}}\}$ ($\mathcal{A}_8$)	✗	0.75	0.82	0.07
	✓	0.68	0.76	0.08
$\ldots, a_{\text{chitChat-public}}\}$ ($\mathcal{A}_9$)	✗	0.61	0.68	0.07
	✓	0.54	0.61	0.07

wording diversity. This is due to the stringent LLM as a judge conditions to ensure quality. Without these quality and diversity constraints, 81% of the story structures are infilled within a single attempt (73% with only quality constraints).

D.3.3 Models Fail Both at Theory of Mind and Pure State Tracking (cont. from § 5.5)

Table D.3: Accuracy breakdown of the experiment shown in Table 5.1, discriminating if each question is *interesting* or not. A question is interesting if the answer would change depending on the entity asked about, thus potentially requiring theory of mind. Results show that part of a model's difficulty with ExploreToM's generated data could be attributed to poor state tracking (i.e., the uninteresting questions, noted ¬Int.).

		Llama			GPT4o			Mixtral		
Action Set	Includes Symmetry?	Acc. Int.	Acc. ¬Int.	% Int.	Acc. Int.	Acc. ¬Int.	% Int.	Acc. Int.	Acc. ¬Int.	% Int.
$\mathcal{A}_1$	✗	0.42	0.20	44%	0.47	0.43	45%	0.44	0.38	48%
	✓	0.31	0.01	49%	0.46	0.28	49%	0.37	0.40	50%
$\mathcal{A}_2$	✗	0.61	0.25	42%	0.57	0.24	41%	0.39	0.01	50%
	✓	0.49	0.22	50%	0.56	0.23	49%	0.19	0.00	34%
$\mathcal{A}_3$	✗	0.54	0.24	47%	0.55	0.35	47%	0.46	0.18	46%
	✓	0.42	0.04	50%	0.57	0.31	48%	0.36	0.04	48%
$\mathcal{A}_4$	✗	0.56	0.11	16%	0.58	0.09	25%	0.49	0.00	25%
	✓	0.25	0.18	49%	0.47	0.21	35%	0.44	0.00	31%
$\mathcal{A}_5$	✗	0.36	0.09	50%	0.50	0.30	45%	0.44	0.40	48%
	✓	0.24	0.15	50%	0.44	0.29	48%	0.44	0.47	50%
$\mathcal{A}_6$	✗	0.37	0.18	50%	0.64	0.25	44%	0.50	0.03	48%
	✓	0.31	0.14	50%	0.64	0.25	41%	0.49	0.04	48%
$\mathcal{A}_7$	✗	0.74	0.74	46%	0.76	0.76	44%	0.58	0.72	42%
	✓	0.74	0.67	43%	0.72	0.75	47%	0.47	0.67	42%
$\mathcal{A}_8$	✗	0.81	0.62	67%	0.80	0.72	66%	0.55	0.42	67%
	✓	0.62	0.59	74%	0.63	0.37	75%	0.41	0.42	74%
$\mathcal{A}_9$	✗	0.48	0.67	40%	0.50	0.42	40%	0.52	0.27	40%
	✓	0.54	0.51	50%	0.65	0.41	50%	0.54	0.27	50%
Total	—	0.49	0.31	48%	0.58	0.37	47%	0.45	0.26	47%

D.3.4 How likely is a randomly-sampled story to require theory of mind? (cont. from §5.5)

Table D.4: Probability that a randomly-sampled story would require theory of mind for answering at least one question. Actions considered are $\{a_{\text{enter}}, a_{\text{leave}}, a_{\text{moveObjContainer}}\}$, all settings of $\{2, 3, 4\}$ people and $\{2, 3, 4\}$ $a_{\text{moveObjContainer}}$ movements, with 10 maximum actions, are shown.

	Number of movements		
Number of people	2	3	4
2	0.131	0.208	0.235
3	0.195	0.234	0.288
4	0.210	0.259	0.315

Table D.5: Probability that a randomly-sampled (story, question) pair would potentially require theory of mind, meaning that the answer to the question varies depending on the entities considered. Actions considered are $\{a_{\text{enter}}, a_{\text{leave}}, a_{\text{moveObjContainer}}\}$, all settings of $\{2, 3, 4\}$ people and $\{2, 3, 4\}$ $a_{\text{moveObjContainer}}$ movements, with 10 maximum actions, are shown.

	Number of movements		
Number of people	2	3	4
2	0.090	0.123	0.124
3	0.120	0.109	0.121
4	0.111	0.101	0.112

Table D.6: Probability that a randomly-sampled (story, question, answer) triple would require an answer that is different from the true world state (i.e., it is a *false-belief question*. Actions considered are $\{a_{\text{enter}}, a_{\text{leave}}, a_{\text{moveObjContainer}}\}$, all settings of $\{2, 3, 4\}$ people and $\{2, 3, 4\}$ $a_{\text{moveObjContainer}}$ movements, with 10 maximum actions, are shown.

	Number of movements		
Number of people	2	3	4
2	0.059	0.084	0.086
3	0.065	0.065	0.072
4	0.056	0.049	0.058

D.3.5 On why a story with a greater number of people may counterintuitively imply a lower average difficulty

In ExploreToM, the number of people and actions is important from a controllability and diversity perspective, but does not directly quantify task difficulty—difficulty quantification for theory of mind is an active area of research. X. A. Huang et al., 2024 quantifies a theory of mind problem complexity as the number of states necessary to solve it correctly (note that their approach requires manual annotation). While the number of states tends to increase with the number of people and actions, many questions do not require analyzing the whole story, e.g. if someone only entered the scene right at the end. When randomly sampling stories while fixing the number of core actions to e.g. 5, it's more likely to have some characters with little involvement in the scene if there are 5 people in total than if there are 2 people. Since accuracy is computed across all questions about all characters, having a larger number of people may bump the average accuracy. ExploreToM's flexible framework allows for minimizing these cases through modifying the lookahead, but we chose against both doing this or filtering questions to show the performance is low even without these considerations.

D.4 Prompts used for generating and validating ExploreToM's data

D.4.1 Generating story contexts (cont. from §5.2.1)

```
Suggest a short context where {num_people} people are together in a room. It should be at most two
sentences long, and they should be able to observe each other. Later in the story, characters are going to
move around and store objects, so your context should be plausible under those constraints. Do not
explicitly include that they can all see each other, it should be clear from context. The room could be in
a house, work environment, etc.

Here's an example for three people. Follow the same format.

LIST CHARACTERS' NAMES:
1. Emily, a meticulous office manager.
2. Jason, a tech-savvy intern.
3. Karen, a diligent accountant.

GIVE SHORT STORY CONTEXT:
Emily, Jason, and Karen gathered around the central table in the sleek office's conference room, discussing
 the upcoming audit. As they strategized, the shelves and storage compartments lining the walls around them
 held the tools and documents they would soon need to organize and pack away.

ROOM IN WHICH THIS STORY BEGINS:

NAME ONE REASONABLE ALTERNATIVE ROOM THEY COULD MOVE TO:

NAME ONE OBJECT TO BE MOVED BY A PERSON DURING THE STORY:

LIST {num_containers} REASONABLE OPAQUE CONTAINERS THAT COULD CONTAIN THIS OBJECT:

LIST {num_topics} DISTINCT AND REASONABLE TOPICS THEY COULD BE CHATTING ABOUT:

To get inspired, make this context happen in {sampled_location}. Suggested names are {sampled_names}, but
feel free to come up with your own names if it would suit the story better. Be direct with your answers: do
 not include parentheses or clarifications beyond the responses requested. Do not refer to plural objects
or give options if a singular thing is requested. The object could be anything--an apple, a pen, a spoon, a
 pair of scissors, a chocolate bar, etc.--, be creative! Avoid vases and microphones, or adding too many
details to the object's description in general.
```

Figure D.2: Prompts used for generating a story context, after infilling the variables (number of people, containers, topics, names, and location). Names and location are sampled independently to increase diversity, prompts shown in Fig. D.3.

```
List 100 names. Do not include any other text.
```

```
Suggest 100 different general contexts in which a story may happen. The context should be able to have
several people in the same location easily listening and observing each other.

1. a school
2. a hospital
3. a vet shop
4. a family living room

Follow the format and make the descriptions as short as possible. Do not include any text before the list.
```

Figure D.3: Prompts used for generating a list of possible characters' names and locations for the story.

D.4.2 Prompts used for story infilling

```
You are an expert writer that uses simple language, avoiding sounding unnatural or cliché. You are clear,
creative, and helpful. You use simple sentence constructions and words so that everyone may understand you.
```

Figure D.4: System prompt used for story infilling

```
Given the following story and knowing the description of the characters involved, write the start of a
story. Don't actually describe any actions in the story, just the setting in which the story will happen.
Only include the characters that are mentioned in the story.

STORY:
{story_script}

CHARACTERS:
{characters_description}

TWO-SENTENCE STORY BEGINNING THAT DOES NOT INCLUDE OR SUGGEST ANY INFORMATION OF WHAT WILL HAPPEN IN THE
STORY. DO NOT MENTION PEOPLE:
```

Figure D.5: System prompt used for sampling narration (the start of the story, before infilling).

```
Given the following story and knowing the description of the characters involved, suggest a reasonable goal
 for each character. Only include the characters that were mentioned in the story.

STORY:
{story_script}

CHARACTERS:
{characters_description}

CHARACTERS GOALS: <insert here>

Follow the format and do not include any other text. Only include the characters mentioned in the story,
and do not even mention the others in your list.
```

Figure D.6: System prompt used for sampling character goals.

```
Continue the story {story_length}, clearly conveying the action or information below without altering it.
Do not contradict any prior information. Avoid repeating the information verbatim, instead naturally (and
possibly implicitly, but still unambiguously) conveying the meaning. Do not add characters or actions that
were not explicitly described. Do not replace characters even if this would improve flow. Combining actions
 into a single sentence is OK as long as you do not alter the original information. {infilling_text_type}

Make it a short, yet an interesting story to read. Make the text exciting to read as well as each character
's speech, so try to avoid e.g. starting all the sentences the same way. The story needs to follow common
sense, e.g. do not magically change an object's location without mentioning it. Do not include any notes,
comments, parentheses, or any other form of extra text that would not belong in a story. Feel free to hint
or describe characters' goals and motivations for performing the actions if it would make the story flow
better.

As a warning, take into account that when someone tells someone privately they might not be in the same
location, e.g. they might be sending a text message or making a phone call; they might also be in the same
location, in that case they could also communicate through a gesture, a whisper, etc. Do not assume a
person is in the same room if it has not been made explicit before. Also, if someone was spying, or if they
 were distracted and did not listen or saw something happen, do not forget to include it! Remember that in
this case, it should be clear that the distraction or spying applies only to the action mentioned and they
go back to normal after the action is finished. Avoid making multiple copies of the same object.

Give {num_tries_completions} responses, ensuring to give {num_tries_completions} different phrasings of
continuing the story conveying the action. Use very different wordings and sentence structures, but avoid
changing object or room names!

WHO ARE THE CHARACTERS: {people_with_personas}

WHAT ARE THEIR GOALS: {optional_characters_goals}

NEW ACTION OR INFORMATION TO INCLUDE: {new_information}

CURRENT SHORT STORY: {story_context}

Follow the format and do not include any other text. Do not include any text before the list. Do not
enumerate. Continue the story {story_length}. Avoid repeating the information verbatim, instead naturally (
and possibly implicitly, but still unambiguously) conveying the meaning.

STORY CONTINUATION: <fill>

STORY CONTINUATION: <fill>
```

Figure D.7: Prompt used for iterative story infilling including characters' goals, and allowing for simultaneous sampling of several possible infillings, which when associated with repetition penalty, yields more diverse infillings. Infilling length is uniformly chosen between *'with a single sentence'* and *'with up to two sentences'*, and infilling text type is uniformly chosen between *'Make the new text be declarative, without including conversations.'* and *'Make the new text conversational, using direct quotes to convey the words spoken by a character.'*

D.5 ExploreToM examples (cont. from §5.3)

See a large sample of ExploreToM-generated data in https://huggingface.co/datasets/facebook/ExploreToM. We also include a few examples below.

D.5.1 Story structure examples

- Addison entered the monastery dining hall.
- Addison filled the large ceramic vase with fresh sunflowers.
- Addison left the monastery dining hall.
- Charlotte entered the monastery dining hall.
- Charlotte painted the large ceramic vase with intricate designs in gold.
- Charlotte glued a few loose diamonds around the neck of the large ceramic vase. While this action was happening, Addison witnessed this action in secret (and only this action).

- Amelia entered the staff room.
- Amelia moved the large first aid kit to the plastic storage bin, which is also located in the staff room. While this action was happening, Alexis witnessed this action in secret (and only this action).
- Amelia entered the equipment storage room.
- Amelia left the equipment storage room.
- Amelia entered the staff room.
- Amelia moved the large first aid kit to the equipment storage room, leaving the plastic storage bin in its original location.
- Amelia moved the large first aid kit to the metal cabinet, which is also located in the equipment storage room.

- Alexander entered the city hall planning department.
- Alexander moved the large map of the city to the cardboard tube, which is also located in the city hall planning department.
- Alexander told privately to Leslie that the large map of the city is in the cardboard tube.
- Victoria entered the city hall planning department. Leslie told privately to Peyton that the large map of the city is in the cardboard tube.
- Victoria moved the large map of the city to the plastic storage bin, which is also located in the city hall planning department.

Figure D.8: Examples of the following story structures: $a_{\text{updateObjState}}$ ($\mathcal{A}_2$) with asymmetry; $\{a_{\text{moveObjContainer}}, a_{\text{moveObjRoom}}\}$ ($\mathcal{A}_4$) with asymmetry; and $\{a_{\text{moveObjContainer}, a_{\text{info}}}\}$ ($\mathcal{A}_5$) without asymmetry.

D.5.2 Story infilling examples

Table D.7: Example of adversarially sampled story structure and iterative infilling with Llama-3.1 70B Instruct, including a story context (first row). This example showcases how the infilling process is able to group sentences when they depict the same event but for more than one person, allowing for an even more natural-sounding story.

Story Structure (list of actions)	Infilling for each action
<initial context>	The operations center, a bustling hub of activity during the park's peak festival season, was dimly lit except for a few overhead fluorescent lights that cast a sterile glow over the rows of workstations and storage units. The air was thick with the smell of stale coffee and fresh printer paper, a familiar scent to anyone who spent time behind the scenes of the busy amusement park.
Brooklyn entered the operations center. James entered the operations center. More text here.	As she entered the operations center, Brooklyn's presence was swiftly followed by James, whose rugged demeanor stood in contrast to the sterile environment of the room.
James moved the large festival schedule poster to the cardboard box, which is also located in the operations center.	James neatly relocated the oversized festival schedule poster, carefully placing it within the confines of a cardboard box, a receptacle that had been securely positioned in a corner of the operations center.
James left the operations center. Brooklyn left the operations center.	Brooklyn stepped out into the bright sunlight, a stark contrast to the dimly lit operations center she had just left, while James followed suit, the warm rays illuminating his weathered features. The festival preparations were now unfolding across the park, with both of them poised to tackle their respective tasks.
Tyler entered the operations center. Brooklyn entered the operations center.	As Tyler walked into the operations center, the soft hum of idle machinery and the faint scent of printer paper enveloped him, a familiar yet calming atmosphere before the chaos of the festival. The room's sterile lighting seemed to intensify, reflecting off the polished surfaces as Brooklyn slipped in, her footsteps quiet on the worn floor.
Brooklyn moved the large festival schedule poster to the plastic storage bin, which is also located in the operations center.	Brooklyn's hands moved swiftly as she carefully slid the oversized festival schedule poster into the large plastic storage bin in the operations center.

Table D.8: Example of adversarially sampled story structure and simple iterative infilling with Llama-3.1 70B Instruct, along with example questions and model answers.

Story Structure (list of actions)	Infilling for each action
David entered the study room. Sarah entered the study room.	David walked into the study room, scanning the organized chaos of files on the desk, just as Sarah stepped in, clutching her tablet and a determined look.
Sarah moved the prototype model to the metal filing cabinet, which is also located in the study room.	Sarah moved the prototype model to the metal filing cabinet, her fingers brushing against the cool surface before turning to David with a confident smile.
David left the study room.	After a brief exchange with Sarah, David grabbed his briefcase and left the study room, mindful of the upcoming meeting downtown.
Mark entered the study room.	Mark entered the study room, his eyes quickly assessing the scattered documents and half-finished sketches on the table, before settling into his favorite chair by the window to brainstorm new ideas.
Mark moved the prototype model to the wooden chest, which is also located in the study room.	Mark, contemplating his latest concept, carefully picked up the prototype model from the metal filing cabinet and transferred it to the wooden chest, a better place for such an inspiration-spurring creation.

Selected Questions	ExploreToM -generated answer using our domain-specific language	GPT-4o's responses	Correct?
In which container will David search for the prototype model?	metal filing cabinet	david will search for the prototype model in the metal filing cabinet.	Yes
In which container will Sarah search for the prototype model?	wooden chest	sarah will search for the prototype model in the metal filing cabinet.	No
In which container will Mark search for the prototype model?	wooden chest	mark will search for the prototype model in the wooden chest.	Yes
In which container will David search for the prototype model?	metal filing cabinet	the metal filing cabinet.	Yes
In which container will Sarah search for the prototype model?	wooden chest	sarah will search for the prototype model in the metal filing cabinet.	No
In which container will Mark search for the prototype model?	wooden chest	the wooden chest.	Yes

References

(FAIR), M. F. A. R. D. T. et al. (2022). "Human-level play in the game of Diplomacy by combining language models with strategic reasoning." In: *Science* 378.6624, pp. 1067–1074. DOI: 10.1126/science.ade9097. eprint: https://www.science.org/doi/pdf/10.1126/science.ade9097. URL: https://www.science.org/doi/abs/10.1126/science.ade9097.

Adhikari, A., Yuan, X., Côté, M.-A., Zelinka, M., Rondeau, M.-A., Laroche, R., Poupart, P., Tang, J., Trischler, A., and Hamilton, W. (2020). "Learning dynamic belief graphs to generalize on text-based games." In: *Advances in Neural Information Processing Systems* 33, pp. 3045–3057.

Aghajanyan, A. (June 2023). "Tweet: `Susan & I found MMLU performance jump 6-10 points in the 40s by formatting multiple choice as (A) not A in MMLU (for internal model). All evaluation of LLM's are broken. Evaluating a task requires marginalizing across all prompts that describe the task, not point estimate of one.`" In: URL: https://twitter.com/ArmenAgha/status/1669084129261162497 (visited on 06/14/2023).

Aghajanyan, A., Yu, L., Conneau, A., Hsu, W.-N., Hambardzumyan, K., Zhang, S., Roller, S., Goyal, N., Levy, O., and Zettlemoyer, L. (23–29 Jul 2023). "Scaling Laws for Generative Mixed-Modal Language Models." In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 265–279. URL: https://proceedings.mlr.press/v202/aghajanyan23a.html.

Ahuja, K., Sclar, M., and Tsvetkov, Y. (2025). "Finding Flawed Fictions: Evaluating Complex Reasoning in Language Models via Plot Hole Detection." In: *Second Conference on Language Modeling*.

Akula, A. R., Wang, K., Liu, C., Saba-Sadiya, S., Lu, H., Todorovic, S., Chai, J., and Zhu, S.-C. (2022). "CX-ToM: Counterfactual explanations with theory-of-mind for enhancing human trust in image recognition models." In: *iScience* 25.1, p. 103581. ISSN: 2589-0042. DOI: https://doi.org/10.1016/j.isci.2021.103581. URL: https://www.sciencedirect.com/science/article/pii/S2589004221015510.

Almazrouei, E. et al. (2023). "Falcon-40B: an open large language model with state-of-the-art performance." In.

Ammanabrolu, P. and Riedl, M. (2021). "Learning Knowledge Graph-based World Models of Textual Environments." In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. URL: https://openreview.net/forum?id=o24k_XfIe6_.

Anil, C., Wu, Y., Andreassen, A. J., Lewkowycz, A., Misra, V., Ramasesh, V. V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. (2022). "Exploring Length Generalization in Large Language Models." In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 38546–38556. URL: https://openreview.net/forum?id=zSkYVeX7bC4.

Arodi, A. and Cheung, J. C. K. (Nov. 2021). "Textual Time Travel: A Temporally Informed Approach to Theory of Mind." In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 4162–4172. DOI: 10.18653/v1/2021.findings-emnlp.351. URL: https://aclanthology.org/2021.findings-emnlp.351.

Bansal, A., Schwarzschild, A., Borgnia, E., Emam, Z., Huang, F., Goldblum, M., and Goldstein, T. (2022). "End-to-end Algorithm Synthesis with Recurrent Networks: Extrapolation without Overthinking." In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., pp. 20232–20242. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/7f70331dbe58ad59d83941dfa7d975aa-Paper-Conference.pdf.

Bara, C.-P., Sky, C.-W., and Chai, J. (2021). "MindCraft: Theory of Mind Modeling for Situated Dialogue in Collaborative Tasks." In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1112–1125.

Baron-Cohen, S., Leslie, A. M., and Frith, U. (1985). "Does the autistic child have a "theory of mind"?" In: *Cognition* 21.1, pp. 37–46.

Baron-Cohen, S., O'Riordan, M., Jones, R., Stone, V., and Plaisted, K. (1999). "A new test of social sensitivity: Detection of faux pas in normal children and children with Asperger syndrome." In: *Journal of Autism and Developmental Disorders* 29.5, pp. 407–418.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). "A neural probabilistic language model." In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press.

Bertsekas, D. (2022). *Abstract Dynamic Programming: 3rd Edition*. Athena scientific optimization and computation series. Athena Scientific. ISBN: 9781886529472. URL: https://books.google.com/books?id=8mhXEAAAQBAJ.

Betz, G., Voigt, C., and Richardson, K. (2021). "Critical Thinking for Language Models." In: *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pp. 63–75.

Bhattamishra, S., Ahuja, K., and Goyal, N. (2020). "On the Practical Ability of Recurrent Neural Networks to Recognize Hierarchical Languages." In: *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. Ed. by D. Scott, N. Bel, and C. Zong. International Committee on Computational Linguistics, pp. 1481–1494. DOI: 10.18653/v1/2020.coling-main.129. URL: https://doi.org/10.18653/v1/2020.coling-main.129.

Białecka-Pikul, M., Kołodziejczyk, A., and Bosacki, S. (2017). "Advanced theory of mind in adolescence: Do age, gender and friendship style play a role?" In: *Journal of Adolescence* 56, pp. 145–156. ISSN: 0140-1971. DOI: https://doi.org/10.1016/j.adolescence.2017.02.009. URL: https://www.sciencedirect.com/science/article/pii/S014019711730026X.

Bian, N., Han, X., Sun, L., Lin, H., Lu, Y., He, B., Jiang, S., and Dong, B. (May 2024). "ChatGPT Is a Knowledgeable but Inexperienced Solver: An Investigation of Commonsense Problem in Large Language Models." In: *Proceedings of the 2024 Joint International*

Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024). Ed. by N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, and N. Xue. Torino, Italia: ELRA and ICCL, pp. 3098–3110. URL: https://aclanthology.org/2024.lrec-main.276/.

Bogin, B., Gupta, S., and Berant, J. (Dec. 2022). "Unobserved Local Structures Make Compositional Generalization Hard." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 2731–2747. URL: https://aclanthology.org/2022.emnlp-main.175.

Brown, T. et al. (2020). "Language models are few-shot learners." In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 1877–1901.

Bubeck, S. et al. (2023). "Sparks of Artificial General Intelligence: Early experiments with GPT-4." In: *CoRR* abs/2303.12712. DOI: 10.48550/arXiv.2303.12712. arXiv: 2303.12712. URL: https://doi.org/10.48550/arXiv.2303.12712.

Bueno, M. C., Gemmell, C., Dalton, J., Lotufo, R., and Nogueira, R. (Dec. 2022). "Induced Natural Language Rationales and Interleaved Markup Tokens Enable Extrapolation in Large Language Models." In: *Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP)*. Ed. by D. Ferreira, M. Valentino, A. Freitas, S. Welleck, and M. Schubotz. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, pp. 17–24. DOI: 10.18653/v1/2022.mathnlp-1.3. URL: https://aclanthology.org/2022.mathnlp-1.3/.

Carney, J., Wlodarski, R., and Dunbar, R. (2014). "Inference or enaction? The impact of genre on the narrative processing of other minds." In: *PloS one* 9.12, e114172.

Chapelle, O. and Li, L. (2011). "An empirical evaluation of thompson sampling." In: *Advances in neural information processing systems* 24.

Chen, T. and Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

Chen, Z. et al. (Aug. 2024). "ToMBench: Benchmarking Theory of Mind in Large Language Models." In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by L.-W. Ku, A. Martins, and V. Srikumar. Bangkok, Thailand: Association for Computational Linguistics, pp. 15959–15983. DOI: 10.18653/v1/2024.acl-long.847. URL: https://aclanthology.org/2024.acl-long.847/.

Chiang, D., Cholak, P., and Pillay, A. (2023). "Tighter Bounds on the Expressivity of Transformer Encoders." In: *Proceedings of the 40th International Conference on Machine Learning*. Proceedings of Machine Learning Research 202. Ed. by A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, pp. 5544–5562. URL: https://proceedings.mlr.press/v202/chiang23a.html.

Choi, J. H., Hickman, K. E., Monahan, A., and Schwarcz, D. (2023). "ChatGPT goes to law school." In: *Available at SSRN*.

Chowdhery, A. et al. (2023). "PaLM: Scaling Language Modeling with Pathways." In: *Journal of Machine Learning Research* 24.240, pp. 1–113. URL: http://jmlr.org/papers/v24/22-1144.html.

Chung, H. W. et al. (2024). "Scaling Instruction-Finetuned Language Models." In: *Journal of Machine Learning Research* 25.70, pp. 1–53. URL: http://jmlr.org/papers/v25/23-0870.html.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). "Better hypothesis testing for statistical machine translation: Controlling for optimizer instability." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 176–181.

Clark, P., Tafjord, O., and Richardson, K. (2021). "Transformers as soft reasoners over language." In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3882–3890.

Creswell, A. and Shanahan, M. (2022). "Faithful Reasoning Using Large Language Models." In: *CoRR* abs/2208.14271. DOI: 10.48550/arXiv.2208.14271. arXiv: 2208.14271. URL: https://doi.org/10.48550/arXiv.2208.14271.

Creswell, A., Shanahan, M., and Higgins, I. (2023). "Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning." In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=3Pf3Wg6o-A4.

Dehaene, S., Molko, N., Cohen, L., and Wilson, A. J. (2004). "Arithmetic and the brain." In: *Current opinion in neurobiology* 14.2, pp. 218–224.

Delyon, B. and Juditsky, A. (1999). "On Small Perturbations of Stable Markov Operators: Unbounded Case." In: *Theory of Probability & Its Applications* 43.4, pp. 577–587. DOI: 10.1137/S0040585X97977173. eprint: https://doi.org/10.1137/S0040585X97977173. URL: https://doi.org/10.1137/S0040585X97977173.

Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E., and Hu, Z. (Dec. 2022). "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 3369–3391. DOI: 10.18653/v1/2022.emnlp-main.222. URL: https://aclanthology.org/2022.emnlp-main.222.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). "GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale." In: *Advances in Neural Information Processing Systems*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. URL: https://openreview.net/forum?id=dXiGWqBoxaD.

Diaconis, P. and Freedman, D. (1999). "Iterated random functions." In: *SIAM review* 41.1, pp. 45–76.

Ding, N., Hu, S., Zhao, W., Chen, Y., Liu, Z., Zheng, H., and Sun, M. (2022). "OpenPrompt: An Open-source Framework for Prompt-learning." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 105–113.

Douc, R., Moulines, E., and Stoffer, D. (2014). *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis. ISBN: 9781466502253. URL: https://books.google.com/books?id=y6SNAgAAQBAJ.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). "The llama 3 herd of models." In: *arXiv preprint arXiv:2407.21783*.

Dubois, Y., Dagan, G., Hupkes, D., and Bruni, E. (2020). "Location Attention for Extrapolation to Longer Sequences." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 403–413.

Duijn, M. J. van, Sluiter, I., and Verhagen, A. (2015). "When narrative takes over: The representation of embedded mindstates in Shakespeare's Othello." In: *Language and Literature* 24.2, pp. 148–166. DOI: 10.1177/0963947015572274. eprint: https://doi.org/10.1177/0963947015572274. URL: https://doi.org/10.1177/0963947015572274.

Dziri*, N., Lu*, X., Sclar*, M., et al. (2023). "Faith and Fate: Limits of Transformers on Compositionality." In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., pp. 70293–70332. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/deb3c28192f979302c157cb653c15e90-Paper-Conference.pdf.

Elazar, Y., Kassner, N., Ravfogel, S., Ravichander, A., Hovy, E., Schütze, H., and Goldberg, Y. (2021). "Measuring and improving consistency in pretrained language models." In: *Transactions of the Association for Computational Linguistics* 9, pp. 1012–1031.

Evans, J. S. B. (1989). *Bias in human reasoning: Causes and consequences*. Lawrence Erlbaum Associates, Inc.

Faghihi, H. R. and Kordjamshidi, P. (2021). "Time-Stamped Language Model: Teaching Language Models to Understand The Flow of Events." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4560–4570.

Feng, H., Lian, L., Dunlap, L., Shu, J., Wang, X., Wang, R., Darrell, T., Suhr, A., and Kanazawa, A. (2025). "Visually Prompted Benchmarks Are Surprisingly Fragile." In: *arXiv preprint arXiv:2512.17875*.

Frith, C., Wolpert, D., Frith, U., and Frith, C. D. (2003). "Development and neurophysiology of mentalizing." In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 358.1431, pp. 459–473. DOI: 10.1098/rstb.2002.1218. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rstb.2002.1218. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2002.1218.

Fu Yao; Peng, H. and Khot, T. (Dec. 2022). "How does GPT Obtain its Ability? Tracing Emergent Abilities of Language Models to their Sources." In: *Yao Fu's Notion*. URL: https://yaofu.notion.site/How-does-GPT-Obtain-its-Ability-Tracing-Emergent-Abilities-of-Language-Models-to-their-Sources-b9a57ac0fcf74f30a1ab9e3e36fa1dc1.

Gandhi, K., Fränken, J.-P., Gerstenberg, T., and Goodman, N. (2024). "Understanding social reasoning in language models with language models." In: *Advances in Neural Information Processing Systems* 36.

Gao, T., Fisch, A., and Chen, D. (Aug. 2021). "Making Pre-trained Language Models Better Few-shot Learners." In: pp. 3816–3830. DOI: 10.18653/v1/2021.acl-long.295. URL: https://aclanthology.org/2021.acl-long.295.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., Peters, M., Schmitz, M., and Zettlemoyer, L. (July 2018). "AllenNLP: A Deep Semantic Natural Language Processing Platform." In: *Proceedings of Workshop for NLP Open Source Software (NLP-*

OSS). Melbourne, Australia: Association for Computational Linguistics, pp. 1–6. DOI: 10.18653/v1/W18-2501. URL: https://aclanthology.org/W18-2501.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). "Shortcut learning in deep neural networks." In: *Nature Machine Intelligence* 2.11, pp. 665–673.

Gonen, H., Iyer, S., Blevins, T., Smith, N., and Zettlemoyer, L. (Dec. 2023). "Demystifying Prompts in Language Models via Perplexity Estimation." In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, pp. 10136–10148. DOI: 10.18653/v1/2023.findings-emnlp.679. URL: https://aclanthology.org/2023.findings-emnlp.679/.

Grant, E., Nematzadeh, A., and Griffiths, T. L. (2017). "How Can Memory-Augmented Neural Networks Pass a False-Belief Task?" In: *Cognitive Science*.

Guarnera, L., Giudice, O., and Battiato, S. (2020). "Deepfake detection by analyzing convolutional traces." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 666–667.

Hahn, M. (2020). "Theoretical limitations of self-attention in neural sequence models." In: *Transactions of the Association for Computational Linguistics* 8, pp. 156–171.

Hamilton, J. D. (1994). *Time series analysis*. Princeton university press.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). "A formal basis for the heuristic determination of minimum cost paths." In: *IEEE transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Helwe, C., Clavel, C., and Suchanek, F. M. (2021). "Reasoning with Transformer-based Models: Deep Learning, but Shallow Reasoning." In: *3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021*. Ed. by D. Chen, J. Berant, A. McCallum, and S. Singh. DOI: 10.24432/C5W300. URL: https://doi.org/10.24432/C5W300.

Henaff, M., Weston, J., Szlam, A., Bordes, A., and LeCun, Y. (2017). "Tracking the World State with Recurrent Entity Networks." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJTKKKqeg.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). "Deep reinforcement learning that matters." In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). "Measuring Massive Multitask Language Understanding." In: *International Conference on Learning Representations*.

Henighan, T. et al. (2020). "Scaling Laws for Autoregressive Generative Modeling." In: *CoRR* abs/2010.14701. arXiv: 2010.14701. URL: https://arxiv.org/abs/2010.14701.

Hiebert, J. (2013). *Conceptual and procedural knowledge: The case of mathematics*. Routledge.

Honovich, O., Shaham, U., Bowman, S. R., and Levy, O. (July 2023). "Instruction Induction: From Few Examples to Natural Language Task Descriptions." In: pp. 1935–1952. DOI: 10.18653/v1/2023.acl-long.108. URL: https://aclanthology.org/2023.acl-long.108.

Hu, S., Zhou, H., Hergul, M., Gritta, M., Zhang, G., Iacobacci, I., Vulić, I., and Korhonen, A. (2023). "Multi 3 woz: A multilingual, multi-domain, multi-parallel dataset for training

and evaluating culturally adapted task-oriented dialog systems." In: *Transactions of the Association for Computational Linguistics* 11, pp. 1396–1415.

Huang, X. A., La Malfa, E., Marro, S., Asperti, A., Cohn, A. G., and Wooldridge, M. J. (Nov. 2024). "A Notion of Complexity for Theory of Mind via Discrete World Models." In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Y. Al-Onaizan, M. Bansal, and Y.-N. Chen. Miami, Florida, USA: Association for Computational Linguistics, pp. 2964–2983. DOI: 10.18653/v1/2024.findings-emnlp.167. URL: https://aclanthology.org/2024.findings-emnlp.167.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. (2023). "Editing Models with Task Arithmetic." In: *International Conference on Learning Representations*.

Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. (2017). "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control." In: *arXiv preprint arXiv:1708.04133*.

Jacqmin, L., Barahona, L. M. R., and Favre, B. (2022). ""Do you follow me?": A Survey of Recent Approaches in Dialogue State Tracking." In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 336–350.

Jansen, P. (2022). "A Systematic Survey of Text Worlds as Embodied Natural Language Environments." In: *The Third Wordplay: When Language Meets Games Workshop*. URL: https://openreview.net/forum?id=wAzgsRT_Uc3.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. (2024). "Mixtral of experts." In: *arXiv preprint arXiv:2401.04088*.

Jiang, A. Q., Welleck, S., Zhou, J. P., Lacroix, T., Liu, J., Li, W., Jamnik, M., Lample, G., and Wu, Y. (2023). "Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs." In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=SMa9EAovKMC.

Jiang, L., Hwang, J. D., Bhagavatula, C., Bras, R. L., Forbes, M., Borchardt, J., Liang, J., Etzioni, O., Sap, M., and Choi, Y. (2021). "Delphi: Towards Machine Ethics and Norms." In: *CoRR* abs/2110.07574. arXiv: 2110.07574. URL: https://arxiv.org/abs/2110.07574.

Jiang, Z., Xu, F. F., Araki, J., and Neubig, G. (2020). "How can we know what language models know?" In: *Transactions of the Association for Computational Linguistics* 8, pp. 423–438.

Jin, C., Wu, Y., Cao, J., Xiang, J., Kuo, Y.-L., Hu, Z., Ullman, T., Torralba, A., Tenenbaum, J., and Shu, T. (Aug. 2024). "MMToM-QA: Multimodal Theory of Mind Question Answering." In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by L.-W. Ku, A. Martins, and V. Srikumar. Bangkok, Thailand: Association for Computational Linguistics, pp. 16077–16102. URL: https://aclanthology.org/2024.acl-long.851.

Johnson-Laird, P. N., Khemlani, S. S., and Goodwin, G. P. (2015). "Logic, probability, and human reasoning." In: *Trends in cognitive sciences* 19.4, pp. 201–214.

Jung, C., Kim, D., Jin, J., Kim, J., Seonwoo, Y., Choi, Y., Oh, A., and Kim, H. (Nov. 2024). "Perceptions to Beliefs: Exploring Precursory Inferences for Theory of Mind in Large Language Models." In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Y. Al-Onaizan, M. Bansal, and Y.-N. Chen.

Miami, Florida, USA: Association for Computational Linguistics, pp. 19794–19809. DOI: 10.18653/v1/2024.emnlp-main.1105. URL: https://aclanthology.org/2024.emnlp-main.1105/.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). "Scaling Laws for Neural Language Models." In: *CoRR* abs/2001.08361. arXiv: 2001.08361. URL: https://arxiv.org/abs/2001.08361.

Kazemi, M., Kim, N., Bhatia, D., Xu, X., and Ramachandran, D. (July 2023). "LAMBADA: Backward Chaining for Automated Reasoning in Natural Language." In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, pp. 6547–6568. DOI: 10.18653/v1/2023.acl-long.361. URL: https://aclanthology.org/2023.acl-long.361/.

Khashabi, D. et al. (July 2022). "Prompt Waywardness: The Curious Case of Discretized Interpretation of Continuous Prompts." In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, pp. 3631–3643. DOI: 10.18653/v1/2022.naacl-main.266. URL: https://aclanthology.org/2022.naacl-main.266.

Kim, H., Sclar, M., Zhou, X., Bras, R., Kim, G., Choi, Y., and Sap, M. (2023). "FANToM: A Benchmark for Stress-testing Machine Theory of Mind in Interactions." In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14397–14413.

Kim, S. et al. (Apr. 2025). "The BiGGen Bench: A Principled Benchmark for Fine-grained Evaluation of Language Models with Language Models." 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 L. Chiruzzo, A. Ritter, and L. Wang. Albuquerque, New Mexico: Association for Computational Linguistics, pp. 5877–5919. ISBN: 979-8-89176-189-6. DOI: 10.18653/v1/2025.naacl-long.303. URL: https://aclanthology.org/2025.naacl-long.303/.

Kinderman, P., Dunbar, R., and Bentall, R. P. (1998). "Theory-of-mind deficits and causal attributions." In: *British journal of Psychology* 89.2, pp. 191–204.

Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*. USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321295358.

Koralus, P. E. and Wang-Mascianica, V. (2023). "Humans in Humans Out: On GPT Converging Toward Common Sense in both Success and Failure." In: *CoRR* abs/2303.17276. DOI: 10.48550/arXiv.2303.17276. arXiv: 2303.17276. URL: https://doi.org/10.48550/arXiv.2303.17276.

Kosinski, M. (2024). "Evaluating large language models in theory of mind tasks." In: *Proceedings of the National Academy of Sciences* 121.45, e2405460121.

Lai, T. L., Robbins, H., et al. (1985). "Asymptotically efficient adaptive allocation rules." In: *Advances in applied mathematics* 6.1, pp. 4–22.

Le, M., Boureau, Y.-L., and Nickel, M. (2019). "Revisiting the evaluation of theory of mind through question answering." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5872–5877.

Leslie, A. M., Friedman, O., and German, T. P. (2004). "Core mechanisms in 'theory of mind'." In: *Trends in cognitive sciences* 8.12, pp. 528–533.

Lester, B., Al-Rfou, R., and Constant, N. (2021). "The Power of Scale for Parameter-Efficient Prompt Tuning." In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059.

Leverage, P., Mancing, H., and Schweickert, R. (2010). *Theory of mind and literature*. Purdue University Press.

Levy, I., Bogin, B., and Berant, J. (July 2023). "Diverse Demonstrations Improve In-context Compositional Generalization." In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, pp. 1401–1422. URL: https://aclanthology.org/2023.acl-long.78.

Li, Q., Cui, L., Zhao, X., Kong, L., and Bi, W. (Aug. 2024). "GSM-Plus: A Comprehensive Benchmark for Evaluating the Robustness of LLMs as Mathematical Problem Solvers." In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by L.-W. Ku, A. Martins, and V. Srikumar. Bangkok, Thailand: Association for Computational Linguistics, pp. 2961–2984. DOI: 10.18653/v1/2024.acl-long.163. URL: https://aclanthology.org/2024.acl-long.163/.

Li, S. S., Sclar, M., Lang, H., Ni, A., He, J., Xu, P., Cohen, A., Park, C. Y., Tsvetkov, Y., and Celikyilmaz, A. (2025). "PrefPalette: Personalized Preference Modeling with Latent Attributes." In: *Second Conference on Language Modeling*.

Li, X. L., Kaiyom, F., Liu, E. Z., Mai, Y., Liang, P., and Hashimoto, T. (2025). "AutoBencher: Towards Declarative Benchmark Construction." In: *The Thirteenth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=ymt4crbbXh.

Liang, Z., Zhang, Z., Bethard, S., and Surdeanu, M. (2023). "Explainable Verbal Reasoner Plus (EVR+): A Natural Language Reasoning Framework that Supports Diverse Compositional Reasoning." In: *arXiv preprint arXiv:2305.00061*.

Lin, C.-Y. (2004). "Rouge: A package for automatic evaluation of summaries." In: *Text summarization branches out*, pp. 74–81.

Lin, C.-C., Jaech, A., Li, X., Gormley, M. R., and Eisner, J. (June 2021). "Limitations of Autoregressive Models and Their Alternatives." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, pp. 5147–5173. DOI: 10.18653/v1/2021.naacl-main.405. URL: https://aclanthology.org/2021.naacl-main.405.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. (2017). "Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 158–167.

Liu, A., Swayamdipta, S., Smith, N. A., and Choi, Y. (Dec. 2022). "WANLI: Worker and AI Collaboration for Natural Language Inference Dataset Creation." In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 6826–6847. URL: https://preview.aclanthology.org/emnlp-22-ingestion/2022.findings-emnlp.508/.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. (2023). "Transformers Learn Shortcuts to Automata." In: *International Conference on Learning Representations*.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. (May 2022). "Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity." In: pp. 8086–8098. DOI: 10.18653/v1/2022.acl-long.556. URL: https://aclanthology.org/2022.acl-long.556.

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., and Jiang, D. (2024). "WizardCoder: Empowering Code Large Language Models with Evol-Instruct." In: URL: https://openreview.net/forum?id=UnUwSIgK5W.

Lupidi, A., Gemmell, C., Cancedda, N., Dwivedi-Yu, J., Weston, J., Foerster, J., Raileanu, R., and Lomeli, M. (2024). "Source2Synth: Synthetic Data Generation and Curation Grounded in Real Data Sources." In: *arXiv preprint arXiv:2409.08239*.

Madaan, A., Hermann, K., and Yazdanbakhsh, A. (2023). "What makes chain-of-thought prompting effective? a counterfactual study." In: *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1448–1535.

Madaan, A. et al. (2023). "Self-Refine: Iterative Refinement with Self-Feedback." In: *Thirty-seventh Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=S37hOerQLB.

Merrill, W. and Sabharwal, A. (2023a). "A Logic for Expressing Log-Precision Transformers." In: *Advances in Neural Information Processing Systems*.

— (2023b). "The Parallelism Tradeoff: Limitations of Log-Precision Transformers." In: *Transactions of the Association for Computational Linguistics* 11, pp. 531–545. DOI: 10.1162/tacl_a_00562. URL: https://aclanthology.org/2023.tacl-1.31.

— (2024). "The Expressive Power of Transformers with Chain of Thought." In: *The Twelfth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=NjNGlPh8Wh.

Mizrahi, M., Kaplan, G., Malkin, D., Dror, R., Shahaf, D., and Stanovsky, G. (2024). "State of What Art? A Call for Multi-Prompt LLM Evaluation." In: *Transactions of the Association for Computational Linguistics* 12, pp. 933–949. DOI: 10.1162/tacl_a_00681. URL: https://aclanthology.org/2024.tacl-1.52/.

Nadeem, M., Bethke, A., and Reddy, S. (2021). "StereoSet: Measuring stereotypical bias in pretrained language models." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5356–5371.

Nematzadeh, A., Burns, K., Grant, E., Gopnik, A., and Griffiths, T. (Oct. 2018). "Evaluating Theory of Mind in Question Answering." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 2392–2400. DOI: 10.18653/v1/D18-1261. URL: https://aclanthology.org/D18-1261.

Newman, B., Hewitt, J., Liang, P., and Manning, C. D. (2020). "The EOS Decision and Length Extrapolation." In: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 276–291.

Nye, M., Tessler, M., Tenenbaum, J., and Lake, B. M. (2021). "Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning." In: *Advances in Neural Information Processing Systems* 34, pp. 25192–25204.

Nye, M. et al. (2022). "Show Your Work: Scratchpads for Intermediate Computation with Language Models." In: *Deep Learning for Code Workshop*. URL: https://openreview.net/forum?id=HBlx2idbkbq.

OpenAI (2022). *ChatGPT: Optimizing language models for dialogue*.

— (2023). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/pdf/2303.08774.pdf.

— (2024). URL: https://openai.com/index/hello-gpt-4o.

Osterhaus, C., Koerber, S., and Sodian, B. (2016). "Scaling of advanced theory-of-mind tasks." In: *Child development* 87.6, pp. 1971–1991.

Pagnoni, A. et al. (July 2025). "Byte Latent Transformer: Patches Scale Better Than Tokens." In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar. Vienna, Austria: Association for Computational Linguistics, pp. 9238–9258. ISBN: 979-8-89176-251-0. DOI: 10.18653/v1/2025.acl-long.453. URL: https://aclanthology.org/2025.acl-long.453/.

Prasad, A., Hase, P., Zhou, X., and Bansal, M. (May 2023). "GrIPS: Gradient-free, Edit-based Instruction Search for Prompting Large Language Models." In: pp. 3845–3864. DOI: 10.18653/v1/2023.eacl-main.277. URL: https://aclanthology.org/2023.eacl-main.277.

Premack, D. and Woodruff, G. (1978). "Does the chimpanzee have a theory of mind?" In: *Behavioral and brain sciences* 1.4, pp. 515–526.

Press, O., Smith, N., and Lewis, M. (2022). "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=R8sQPpGCv0.

Pryzant, R., Iter, D., Li, J., Lee, Y., Zhu, C., and Zeng, M. (Dec. 2023). "Automatic Prompt Optimization with "Gradient Descent" and Beam Search." In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, pp. 7957–7968. DOI: 10.18653/v1/2023.emnlp-main.494. URL: https://aclanthology.org/2023.emnlp-main.494/.

Qin, C., Zhang, A., Zhang, Z., Chen, J., Yasunaga, M., and Yang, D. (Dec. 2023). "Is ChatGPT a General-Purpose Natural Language Processing Task Solver?" In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, pp. 1339–1384. DOI: 10.18653/v1/2023.emnlp-main.85. URL: https://aclanthology.org/2023.emnlp-main.85/.

Qin, G. and Eisner, J. (2021). "Learning How to Ask: Querying LMs with Mixtures of Soft Prompts." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Qiu, L., Zhao, Y., Liang, Y., Lu, P., Shi, W., Yu, Z., and Zhu, S.-C. (Sept. 2022). "Towards Socially Intelligent Agents with Mental State Transition and Human Value." In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Edinburgh, UK: Association for Computational Linguistics, pp. 146–158. URL: https://aclanthology.org/2022.sigdial-1.16.

Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. A., and Botvinick, M. (2018). "Machine theory of mind." In: *International conference on machine learning*. PMLR, pp. 4218–4227.

Rae, J. W. et al. (2021). "Scaling Language Models: Methods, Analysis & Insights from Training Gopher." In: *ArXiv* abs/2112.11446.

Razeghi, Y., Logan IV, R. L., Gardner, M., and Singh, S. (Dec. 2022). "Impact of Pretraining Term Frequencies on Few-Shot Numerical Reasoning." In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 840–854. URL: https://aclanthology.org/2022.findings-emnlp.59.

Romanou, A., Ibrahim, M., Ross, C., Shaib, C., Okta, K., Bell, S., Ovalle, E., Dodge, J., Bosselut, A., Sinha, K., et al. (2026). "Brittlebench: Quantifying LLM robustness via prompt sensitivity." In: *arXiv preprint arXiv:2603.13285*.

Sap, M., LeBras, R., Fried, D., and Choi, Y. (Dec. 2022). "Neural theory-of-mind? on the limits of social intelligence in large lms." In: *Proceedings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 3762–3780. URL: https://preview.aclanthology.org/emnlp-22-ingestion/2022.emnlp-main.248.

Saparov, A. and He, H. (2023). "Language Models Are Greedy Reasoners: A Systematic Formal Analysis of Chain-of-Thought." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=qFVVBzXxR2V.

Schick, T., Udupa, S., and Schütze, H. (2021). "Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp." In: *Transactions of the Association for Computational Linguistics* 9, pp. 1408–1424.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. (2021). "Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks." In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., pp. 6695–6706.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A. (2024). "Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting." In: *The Twelfth International Conference on Learning Representations*.

Sclar, M., Dwivedi-Yu, J., Fazel-Zarandi, M., Tsvetkov, Y., Bisk, Y., Choi, Y., and Celikyilmaz, A. (2025). "Explore Theory of Mind: program-guided adversarial data generation for theory of mind reasoning." In: *The Thirteenth International Conference on Learning Representations*.

Sclar, M., Kumar, S., West, P., Suhr, A., Choi, Y., and Tsvetkov, Y. (July 2023). "Minding Language Models' (Lack of) Theory of Mind: A Plug-and-Play Multi-Character Belief Tracker." In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, pp. 13960–13980. DOI: 10.18653/v1/2023.acl-long.780. URL: https://aclanthology.org/2023.acl-long.780.

Sclar, M., Neubig, G., and Bisk, Y. (17–23 Jul 2022). "Symmetric Machine Theory of Mind." In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato. Vol. 162.

Proceedings of Machine Learning Research. PMLR, pp. 19450–19466. URL: https://proceedings.mlr.press/v162/sclar22a.html.

Shamay-Tsoory, S. G., Harari, H., Aharon-Peretz, J., and Levkovitz, Y. (2010). "The role of the orbitofrontal cortex in affective theory of mind deficits in criminal offenders with psychopathic tendencies." In: *Cortex* 46.5, pp. 668–677.

Shapira, N., Levy, M., Alavi, S. H., Zhou, X., Choi, Y., Goldberg, Y., Sap, M., and Shwartz, V. (Mar. 2024). "Clever Hans or Neural Theory of Mind? Stress Testing Social Reasoning in Large Language Models." In: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Y. Graham and M. Purver. St. Julian's, Malta: Association for Computational Linguistics, pp. 2257–2273. DOI: 10.18653/v1/2024.eacl-long.138. URL: https://aclanthology.org/2024.eacl-long.138/.

Shapira, N., Zwirn, G., and Goldberg, Y. (2023). "How well do large language models perform on faux pas tests?" In: *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10438–10451.

Shi, H., Ye, S., Fang, X., Jin, C., Isik, L., Kuo, Y.-L., and Shu, T. (2025). "MuMA-ToM: multi-modal multi-agent theory of mind." In: *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI'25/IAAI'25/EAAI'25. AAAI Press. ISBN: 978-1-57735-897-8. DOI: 10.1609/aaai.v39i2.32142. URL: https://doi.org/10.1609/aaai.v39i2.32142.

Shi, W., Han, X., Gonen, H., Holtzman, A., Tsvetkov, Y., and Zettlemoyer, L. (Dec. 2023). "Toward Human Readable Prompt Tuning: Kubrick's The Shining is a good movie, and a good prompt too?" In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, pp. 10994–11005. DOI: 10.18653/v1/2023.findings-emnlp.733. URL: https://aclanthology.org/2023.findings-emnlp.733/.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. (Nov. 2020). "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts." In: pp. 4222–4235. DOI: 10.18653/v1/2020.emnlp-main.346. URL: https://aclanthology.org/2020.emnlp-main.346.

Shojaee, P., Mirzadeh, S. I., Alizadeh, K., Horton, M., Bengio, S., and Farajtabar, M. (2025). "The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity." In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Simon, H. A. and Newell, A. (1971). "Human problem solving: The state of the theory in 1970." In: *American psychologist* 26.2, p. 145.

Simon, H. A. (1962). "The Architecture of Complexity." In: *Proceedings of the American Philosophical Society* 106.6, pp. 467–482. ISSN: 0003049X. URL: http://www.jstor.org/stable/985254 (visited on 03/13/2023).

Srivastava, A. et al. (2023). "Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models." In: *Transactions on Machine Learning Research*. Featured Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=uyTL5Bvosj.

Stanovsky, G., Michael, J., Zettlemoyer, L., and Dagan, I. (2018). "Supervised open information extraction." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 885–895.

Stenning, K. and Van Lambalgen, M. (2012). *Human reasoning and cognitive science*. MIT Press.

Strachan, J. W., Albergo, D., Borghini, G., Pansardi, O., Scaliti, E., Gupta, S., Saxena, K., Rufo, A., Panzeri, S., Manzi, G., et al. (2024). "Testing theory of mind in large language models and humans." In: *Nature Human Behaviour*, pp. 1–11.

Su, J., Zhang, J., Ullrich, K., Bottou, L., and Ibrahim, M. (2025). "A single character can make or break your LLM evals." In: *arXiv preprint arXiv:2510.05152*.

Tafjord, O. and Clark, P. (2021). "General-purpose question-answering with macaw." In: *arXiv preprint arXiv:2109.02593*.

Tafjord, O., Dalvi, B., and Clark, P. (2021). "ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634.

Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. (2022). "Galactica: A Large Language Model for Science." In: *CoRR* abs/2211.09085. DOI: 10.48550/arXiv.2211.09085. arXiv: 2211.09085. URL: https://doi.org/10.48550/arXiv.2211.09085.

Thoppilan, R. et al. (2022). "LaMDA: Language Models for Dialog Applications." In: *CoRR* abs/2201.08239. arXiv: 2201.08239. URL: https://arxiv.org/abs/2201.08239.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023a). "Llama 2: Open foundation and fine-tuned chat models." In: *arXiv preprint arXiv:2307.09288*.

Touvron, H. et al. (2023b). "LLaMA: Open and Efficient Foundation Language Models." In: *CoRR* abs/2302.13971. DOI: 10.48550/arXiv.2302.13971. arXiv: 2302.13971. URL: https://doi.org/10.48550/arXiv.2302.13971.

Ullman, T. (2023). "Large Language Models Fail on Trivial Alterations to Theory-of-Mind Tasks." In: *arXiv preprint arXiv:2302.08399*.

Valle, A., Massaro, D., Castelli, I., and Marchetti, A. (2015). "Theory of mind development in adolescence and early adulthood: The growing complexity of recursive thinking ability." In: *Europe's journal of psychology* 11.1, p. 112.

Valmeekam, K., Olmo, A., Sreedharan, S., and Kambhampati, S. (2022). "Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change)." In: *NeurIPS 2022 Foundation Models for Decision Making Workshop*. URL: https://openreview.net/forum?id=wUU-7XTL5XO.

Van Breugel, B., Qian, Z., and Van Der Schaar, M. (2023). "Synthetic data, real errors: how (not) to publish and use synthetic data." In: *International Conference on Machine Learning*. PMLR, pp. 34793–34808.

Veličković, P. and Blundell, C. (2021). "Neural algorithmic reasoning." In: *Patterns* 2.7, p. 100273.

Wang, Q., Saha, K., Gregori, E., Joyner, D., and Goel, A. (2021). "Towards mutual theory of mind in human-ai interaction: How language reflects what students perceive about a

virtual teaching assistant." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–14.

Wang, T., Kulikov, I., Golovneva, O., Yu, P., Yuan, W., Dwivedi-Yu, J., Pang, R. Y., Fazel-Zarandi, M., Weston, J., and Li, X. (2024). "Self-Taught Evaluators." In: *arXiv preprint arXiv:2408.02666*.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (July 2023). "Self-Instruct: Aligning Language Models with Self-Generated Instructions." In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, pp. 13484–13508. DOI: 10.18653/v1/2023.acl-long.754. URL: https://aclanthology.org/2023.acl-long.754.

Wang, Y. et al. (Dec. 2022a). "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks." In: pp. 5085–5109. DOI: 10.18653/v1/2022.emnlp-main.340. URL: https://aclanthology.org/2022.emnlp-main.340.

Wang, Y., zhong, fangwei, Xu, J., and Wang, Y. (2022b). "ToM2C: Target-oriented Multi-agent Communication and Cooperation with Theory of Mind." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=2t7CkQXNpuq.

Wei, A., Haghtalab, N., and Steinhardt, J. (2023). "Jailbroken: How Does LLM Safety Training Fail?" In: *Thirty-seventh Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=jA235JGM09.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., b. ichter brian, Xia, F., Chi, E., Le, Q. V., and Zhou, D. (2022a). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 24824–24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Wei, J. et al. (2022b). "Emergent Abilities of Large Language Models." In: *Transactions on Machine Learning Research*. Survey Certification. ISSN: 2835-8856. URL: https://openreview.net/forum?id=yzkSU5zdwD.

Welleck, S., Liu, J., Lu, X., Hajishirzi, H., and Choi, Y. (2022a). "NaturalProver: Grounded Mathematical Proof Generation with Language Models." In: *Advances in Neural Information Processing Systems*. Vol. 35. Curran Associates, Inc., pp. 4913–4927. URL: https://openreview.net/forum?id=rhdfTOiXBng.

Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. (2023). "Generating Sequences by Learning to Self-Correct." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=hH36JeQZDaO.

Welleck, S., West, P., Cao, J., and Choi, Y. (2022b). "Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 8, pp. 8629–8637.

Wellman, H. M. (2014). *Making minds: How theory of mind develops*. Oxford University Press.

West, P., Bhagavatula, C., Hessel, J., Hwang, J., Jiang, L., Le Bras, R., Lu, X., Welleck, S., and Choi, Y. (July 2022). "Symbolic Knowledge Distillation: from General Language Models to Commonsense Models." In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies. Ed. by M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz. Seattle, United States: Association for Computational Linguistics, pp. 4602–4625. DOI: 10.18653/v1/2022.naacl-main.341. URL: https://aclanthology.org/2022.naacl-main.341.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). "A prompt pattern catalog to enhance prompt engineering with chatgpt." In: *arXiv preprint arXiv:2302.11382*.

Wilf, A., Lee, S., Liang, P. P., and Morency, L.-P. (2024). "Think twice: Perspective-taking improves large language models' theory-of-mind capabilities." In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8292–8308.

Wimmer, H. and Perner, J. (1983). "Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception." In: *Cognition* 13.1, pp. 103–128.

Wood, E., Baltrušaitis, T., Hewitt, C., Dziadzio, S., Cashman, T. J., and Shotton, J. (2021). "Fake it till you make it: face analysis in the wild using synthetic data alone." In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3681–3691.

Wu, Y., He, Y., Jia, Y., Mihalcea, R., Chen, Y., and Deng, N. (Dec. 2023). "Hi-ToM: A Benchmark for Evaluating Higher-Order Theory of Mind Reasoning in Large Language Models." In: ed. by H. Bouamor, J. Pino, and K. Bali, pp. 10691–10706. DOI: 10.18653/v1/2023.findings-emnlp.717. URL: https://aclanthology.org/2023.findings-emnlp.717.

Xu, H., Zhao, R., Zhu, L., Du, J., and He, Y. (2024). "OpenToM: A Comprehensive Benchmark for Evaluating Theory-of-Mind Reasoning Capabilities of Large Language Models." In: *arXiv preprint arXiv:2402.06044*.

Yang, K., Tian, Y., Peng, N., and Klein, D. (Dec. 2022). "Re3: Generating Longer Stories With Recursive Reprompting and Revision." In: ed. by Y. Goldberg, Z. Kozareva, and Y. Zhang, pp. 4393–4479. DOI: 10.18653/v1/2022.emnlp-main.296. URL: https://aclanthology.org/2022.emnlp-main.296.

Yu, L., Jiang, W., Shi, H., YU, J., Liu, Z., Zhang, Y., Kwok, J., Li, Z., Weller, A., and Liu, W. (2024). "MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models." In: URL: https://openreview.net/forum?id=N8N0hgNDRt.

Yu, P., Wang, T., Golovneva, O., AlKhamissi, B., Verma, S., Jin, Z., Ghosh, G., Diab, M., and Celikyilmaz, A. (July 2023). "ALERT: Adapt Language Models to Reasoning Tasks." In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, pp. 1055–1081. DOI: 10.18653/v1/2023.acl-long.60. URL: https://aclanthology.org/2023.acl-long.60/.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. (2022). "Star: Bootstrapping reasoning with reasoning." In: *Advances in Neural Information Processing Systems* 35, pp. 15476–15488.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). "BERTScore: Evaluating Text Generation with BERT." In: *International Conference on Learning Representations*.

Zhang, Y., Backurs, A., Bubeck, S., Eldan, R., Gunasekar, S., and Wagner, T. (2022). "Unveiling Transformers with LEGO: a synthetic reasoning task." In: *CoRR* abs/2206.04301. DOI: 10.48550/arXiv.2206.04301. arXiv: 2206.04301. URL: https://doi.org/10.48550/arXiv.2206.04301.

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., et al. (2025). "Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models: Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models"." In: *Computational Linguistics* 51.4.

Zheng, H. and Zhan, H. (2023). "ChatGPT in scientific writing: a cautionary tale." In: *The American Journal of Medicine*.

Zhou, H., Nova, A., Courville, A., Larochelle, H., Neyshabur, B., and Sedghi, H. (2023). *Teaching Algorithmic Reasoning via In-context Learning*. URL: https://openreview.net/forum?id=6dlC7E1H_9.

Zhou, P., Madaan, A., Potharaju, S. P., Gupta, A., McKee, K. R., Holtzman, A., Pujara, J., Ren, X., Mishra, S., Nematzadeh, A., et al. (2023). "How FaR Are Large Language Models From Agents with Theory-of-Mind?" In: *arXiv preprint arXiv:2310.03051*.

Zhou, P., Zhu, A., Hu, J., Pujara, J., Ren, X., Callison-Burch, C., Choi, Y., and Ammanabrolu, P. (2022). "An AI Dungeon Master's Guide: Learning to Converse and Guide with Intents and Theory-of-Mind in Dungeons and Dragons." In: *arXiv preprint arXiv:2212.10060*.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2023). "Large Language Models are Human-Level Prompt Engineers." In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=92gvk82DE-.

Zhu, H., Neubig, G., and Bisk, Y. (2021). "Few-shot Language Coordination by Modeling Theory of Mind." In: *International Conference on Machine Learning*. PMLR, pp. 12901–12911.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. (2023). "Universal and transferable adversarial attacks on aligned language models." In: *arXiv preprint arXiv:2307.15043*.

Zunshine, L. (2006). *Why we read fiction: Theory of mind and the novel*. Ohio State University Press.