In-context Learning Generalizes, But Not Always Robustly: The Case of Syntax

Anonymous ACL submission

Abstract

In-context learning (ICL) is now a common method for teaching large language models (LLMs) new tasks: given labeled examples in the input context, the LLM learns to perform the task without weight updates. Do models guided via ICL infer the underlying structure of the task defined by the context, or do they rely on superficial heuristics that only generalize to identically distributed examples? We address this question using transformations tasks that assess sensitivity to syntax—a requirement for robust language understanding. We further investigate whether out-of-distribution generalization can be improved via chain-of-thought prompting, where the model is provided with a sequence of intermediate computation steps that illustrate how the task ought to be performed. In experiments with models from the GPT, PaLM, and Llama 2 families, we find large variance across LMs. The variance is explained more by the composition of the pre-training corpus and supervision methods than by model size; in particular, models pre-trained on code generalize better, and benefit more from chain-of-thought prompting.

1 Introduction

Language models (LMs) have become increasingly important subjects of study due to their expressive power and performance at scale. When training large language models (LLMs) on massive amounts of text, surprisingly sophisticated linguistic behaviors, such as in-context learning (ICL), emerge (Brown et al., 2020; Min et al., 2022a): given only a small number of labeled training examples in the input context, LLMs can generalize to new instances of the task without weight updates. Thus, even if we do not have access to the model's weights, we can have an LLM perform new tasks with significantly higher-than-chance performance. Given ICL's reliance on a small number of instances, the question arises as to how much it teaches LLMs about the underlying structure of a task, as opposed to superficial heuristics that do not fully capture the task. In this study, we ask: How robust is ICL to distribution shifts between in-context exemplars and test examples?

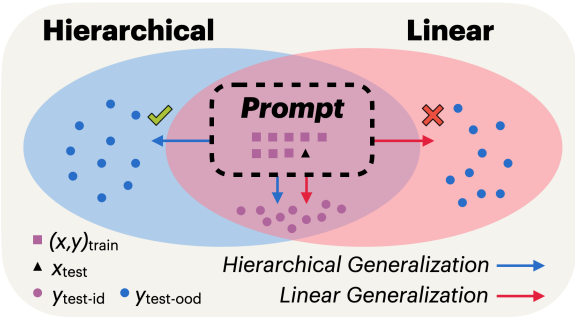


Figure 1: The syntactic transformations paradigm. We prompt language models with labeled examples $(x, y)_{\text{train}}$ that can be explained using either robust syntactic/hierarchical features or spurious positional/linear features. We also include the input from a test example x_{test}. We ensure the models have learned the task by evaluating on in-distribution examples $(x, y)_{\text{test-id}}$. Then, we observe whether models generalize syntactically or linearly on out-of-distribution examples $(x, y)_{\text{test-ood}}$.

We investigate these questions using the test case of syntactic generalization. Accurate syntactic representations are necessary for robust language understanding. In LMs, syntax acquisition is causally associated with significant and abrupt drops in loss and improved performance on NLP tasks (Chen et al., 2023). Language is structured hierarchically, but the structure of sentences is not provided to models as part of their input. Therefore, a model could incorrectly assume that sentences have *linear* rather than *hierarchical* structure—e.g., that syntactic dependencies consist of adjacent words—and thus give an incorrect or unrelated answer to questions such as the following:

(1) Is it true that **children** who don't like **sweets are** rare?

The **main subject** of the question is "**children**", but a model that assumes sentences are structured linearly could instead assume that "**sweets**" is the subject, as it is the positionally closest noun to the **main verb**. In downstream tasks, such assumptions about sentence structure would lead to incorrect responses to inputs that humans find straightforward (McCoy et al., 2019). Recent work using syntactic transformations tasks (Frank and Mathis, 2007) has found that pre-training on large corpora imparts syntactic inductive biases to LMs (Mueller et al., 2022), and the strength of syntactic preferences increases with model depth (Mueller and Linzen, 2023). These studies relied on task-specific fine-tuning, where the LLM's weights are updated based on examples of the task; here, we extend these experiments to the ICL setup.

We investigate LLMs' generalization in two settings: one where they are provided only with a handful of instances of the task, and another in which they are additionally provided with chain-of-thought (CoT) traces (Wei et al., 2022b). CoT refers to the finding that LMs' performance on tasks can be improved if they are instructed to predict the intermediate computations required for the task. We examine whether giving LMs access to syntactic reasoning traces—i.e., to meta-linguistic information—increases their reliance on syntactic features in novel settings. We evaluate not only the accuracy of the final answer, but also the *correctness* of the model's reasoning and the *faithfulness* of the model's final answer to its own reasoning.

We find that, while all models perform well on in-distribution examples, models vary significantly in their ability to generalize out-of-distribution. Surprisingly, even very large LMs are still prone to relying on surface heuristics. We conclude that language models' ICL abilities are limited in tasks where the training and test sets are not identically distributed, and present evidence that exposure to source code during pre-training assists models in overcoming these limitations.

2 Experimental Setup

We use syntactic transformation tasks, where the training and test instances are drawn from distributions that are distinct in controlled ways (§2.1). We test the syntactic generalization of multiple families of LMs (§2.2) when prompted using in-context learning (§2.3).

2.1 Syntactic Transformations

Question formation. Here, a model is given a declarative sentence and must transform it into a yes/no question by moving the main auxiliary verb to the start of the sentence. For instance, given this training example:

(2) The **yaks** near my salamanders **have** amused your unicorn.

The model should move the main auxiliary verb "have" to the start of the sentence to form the question, "**Have** the yaks near my salamanders amused your unicorn?". The model should rely on hierarchical syntactic information s: it should detect the *main auxiliary* in the sentence and move it to the front (the MOVE-MAIN hypothesis). However, the model could instead learn the positional heuristic p and move the *linearly first* auxiliary in the sentence (the MOVE-FIRST hypothesis). Both s and p produce correct outputs for the training examples, but only s generalizes correctly to out-of-distribution inputs where, crucially, the main auxiliary is *not* linearly first in the sentence:

(3) My **zebras** that **have** admired the newt **haven't** observed the peacocks.
  a. ↳MOVE-MAIN: **Haven't** my **zebras** that **have** admired the newt observed the peacocks?
  b. ↳MOVE-FIRST: ***Have** my **zebras** that admired the newt **haven't** observed the peacocks?

Tense reinflection. Here, the task is to convert past-tense verbs into present-tense verbs. We want the model to detect the subject of each verb and reinflect it based on its *subject*'s number s (the AGREE-SUBJECT hypothesis). Here, the training distribution consists of examples where all nouns have the same grammatical number:

(4) Her **newt** around your **unicorn confused** some quail.

Because all nouns have the same number, a model could learn to correctly convert "**confused**" to "**confuses**" by simply agreeing the verbs with the *closest* noun p (the AGREE-RECENT hypothesis). To evaluate which hypothesis the model has learned, we evaluate on examples where only AGREE-SUBJECT produces correct outputs:

Model	Params (est.)	Pre-train Tokens (est.)	% Code (est.)	Additional Training
GPT-3 `text-davinci-001`	175B	400B	?	Fine-tuned on human demonstrations and highly rated model outputs
GPT-3.5 `code-davinci-002`	?	500B	20%	–
GPT-3.5 (Turbo) `gpt-3.5-turbo-0301`	?	500B+	?	?
GPT-3.5 `text-davinci-002`	?	500B+	?	Fine-tuned on human demonstrations and highly rated model outputs
GPT-3.5 `text-davinci-003`	?	500B+	?	Reinforcement learning on human feedback
GPT-4 `gpt-4-0314`	?	?	?	?
PaLM	540B	780B	5%	–
Flan-PaLM	540B	782B	5%	Fine-tuned on human demonstrations
Llama 2	70B	2T	4%	–
CodeLlama	34B	2.5T	20%	Fine-tuned on human and model-generated demonstrations

Table 1: Models used in this study, their estimated number of parameters and pre-training tokens, and the proportion of the pre-training corpus estimated to be source code. We use the largest sizes of each PaLM and (Code)Llama model, as larger models are typically better able to leverage in-context guidance (Wei et al., 2022a).

(5) The **yak** upon my **ravens entertained** her zebras.

a. ↳AGREE-SUBJECT: The **yak** upon my **ravens entertains** her zebras.

b. ↳AGREE-RECENT: *The **yak** upon my **ravens entertain** her zebras.

Task formulation. Each syntactic transformation example (x, y) consists of input sentence x and output sentence y, where x and y are semantically and lexically nearly identical but differ in the syntactic arrangement of the words (as described above).

There is a training set $\mathcal{S}_{\text{train}}$ and two test sets: an in-distribution (ID) test set $\mathcal{S}_{\text{test-id}}$ used to determine whether the model has learned to perform the task, and one out-of-distribution (OOD) test set $\mathcal{S}_{\text{test-ood}}$ used to determine whether the model generalizes in a manner consistent with the latent hierarchical structure of language. The distributions of $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test-ood}}$ differ in controlled ways: the training examples $\mathcal{S}_{\text{train}}$ could be correctly transformed using either syntactic feature s or positional feature p, whereas the test examples *require* reliance on s for correct answers. In other words:

$$\mathcal{S}_{\text{train}} = \{(x, y) \mid s(x) = 1 \wedge p(x) = 1\}$$
$$\mathcal{S}_{\text{test-id}} = \{(x, y) \mid s(x) = 1 \wedge p(x) = 1\}$$
$$\mathcal{S}_{\text{test-ood}} = \{(x, y) \mid s(x) = 1 \wedge p(x) = 0\}$$

See Figure 1 for an illustration. We prompt models with up to 8 exemplars $\{(x_1, y_1)_{\text{train}}, \ldots, (x_8, y_8)_{\text{train}}\}$, followed by the input of an example from one of the test sets x_{test}. A model that has learned to rely only on the spurious feature p will obtain 0% accuracy on $\mathcal{S}_{\text{test-ood}}$, but 100% on $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test-id}}$; only reliance on the syntactic feature s will yield 100% accuracy on $\mathcal{S}_{\text{test-ood}}$.

2.2 Models

We use a series of Transformer-based (Vaswani et al., 2017) decoder-only autoregressive language models that are known to support in-context learning. Estimates of model sizes and pre-training corpus sizes are presented in Table 1. Estimates are derived from the papers that originally described the methods for each model, but as OpenAI does not provide official parameter counts or training set descriptions,[1] the true numbers for GPT models may differ.

The OpenAI GPT models we use include GPT-3 `text-davinci-001` (Brown et al., 2020; Ouyang et al., 2022), GPT-3.5,[2] and GPT-4.[3] The GPT-3.5 variants we use include `code-davinci-002` (Chen et al., 2021), which is pre-trained on natural language and a large amount of source code; `text-davinci-002`, a `code-davinci-002` model which is fine-tuned on instructions and human-labeled examples from many tasks (Wei et al., 2022a); and `text-davinci-003`, which is pre-trained similarly to `text-davinci-002` and then trained via reinforcement learning on human feedback (RLHF; Ouyang et al., 2022). GPT-3.5 (Turbo) builds on `text-davinci-003`, optimizing it for chat. Each of these models is estimated to be pre-trained on 400B or more tokens of text, as each is based on Brown et al. (2020); the models based on `code-davinci-002` are additionally estimated to be pre-trained on at least 100B tokens of

[1] The information OpenAI has released about their models may be found here: https://platform.openai.com/docs/model-index-for-researchers

[2] https://platform.openai.com/docs/models/gpt-3-5

[3] https://platform.openai.com/docs/models/gpt-4

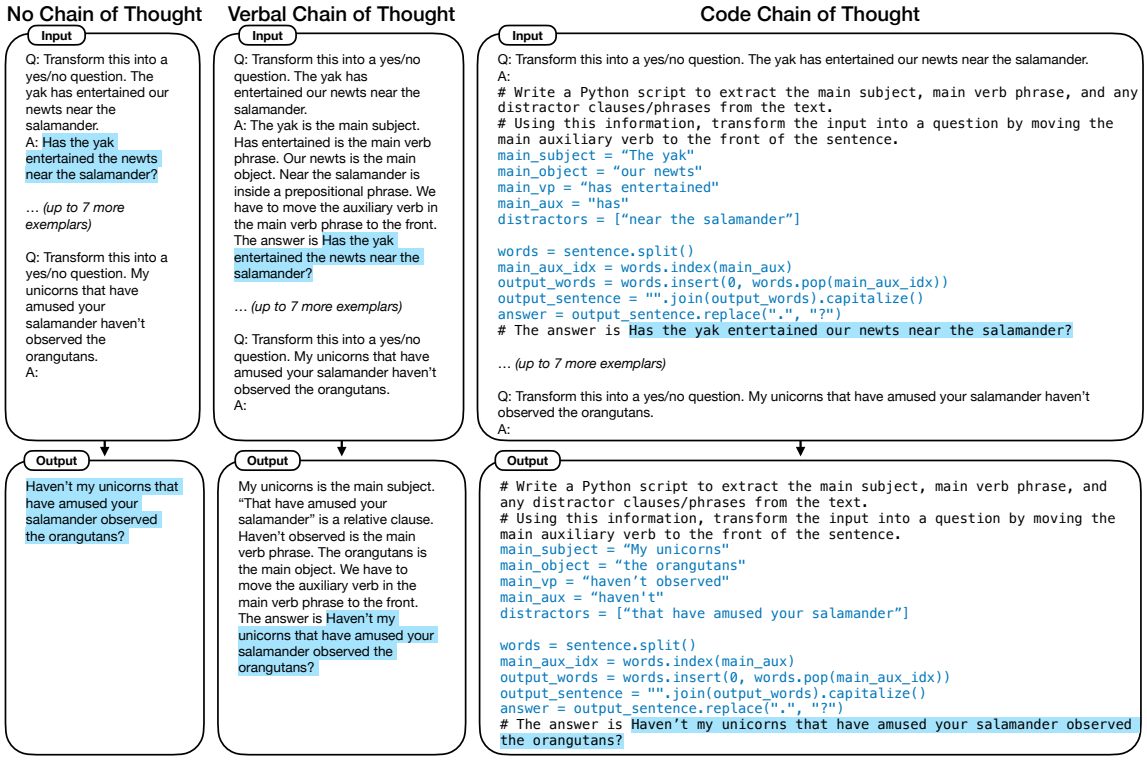


Figure 2: The prompt formats we use for question formation. We give the model up to 8 exemplars, followed by a test example. The model must generate the chain-of-thought reasoning *and* the answer. We highlight the answers with a blue background. In the code prompt, we put code (except comments) in `blue text`. We construct similar prompts for tense reinflection; see Appendix A.

code (Chen et al., 2021). Few details of GPT-4's architecture or pre-training have been made public.

PaLM models (Chowdhery et al., 2022) are pre-trained on large amounts of English text and code; 5% of the pre-training corpus for PaLM is source code. The variants we test have 540B parameters. Flan-PaLM (Chung et al., 2022) is further trained using instruction-finetuning (Wei et al., 2022a).

Finally, we evaluate Llama 2 models (Touvron et al., 2023), which are among the few open-source LMs capable of in-context learning. We use the 70B-parameter model,[4] which is trained on 2T tokens of mostly natural language and an unspecified amount of code. We also use CodeLlama (Rozière et al., 2023), which is Llama 2 trained on an additional 500B tokens of code. In preliminary experiments, we found that the 34B-parameter instruction-tuned variant of CodeLlama[5] performed best, so we use that variant here.

For OpenAI and PaLM models, we use greedy decoding and set the temperature to 0 to reduce variance. For Llama models, we use settings from Touvron et al. (2023): nucleus sampling with $p = 0.9$ and temperature 0.1.[6]

2.3 Prompt Formats

In ICL, a set of labeled examples, concatenated into a "prompt", is provided in the model's context. We use 8 labeled examples (henceforth, *exemplars*). Question formation exemplars are drawn from the training portion of Mueller et al. (2022). Tense reinflection exemplars are drawn from the training portion of McCoy et al. (2020).

We also experiment with two types of manually written CoT traces, which explicitly provide cues to the syntactic structure of the exemplars; these are provided between the input sequence and the answer. The Verbal CoT prompt includes verbal descriptions of the syntactic structure of the sentence, where the model must describe which words compose the main subject, the main verb phrase, and other components. The Code CoT prompt like-

[4] For Llama 2 (70B), we use 4-bit quantization so that the model will fit on a single GPU.

[5] `codellama/CodeLlama-34b-Instruct-hf`

[6] We use different decoding hyperparameters because replicability is more crucial for models that are more difficult and costly to access. For Llama models, we consider slightly higher variance to be more acceptable in exchange for higher expected performance, for replication and trying other settings are both more accessible.

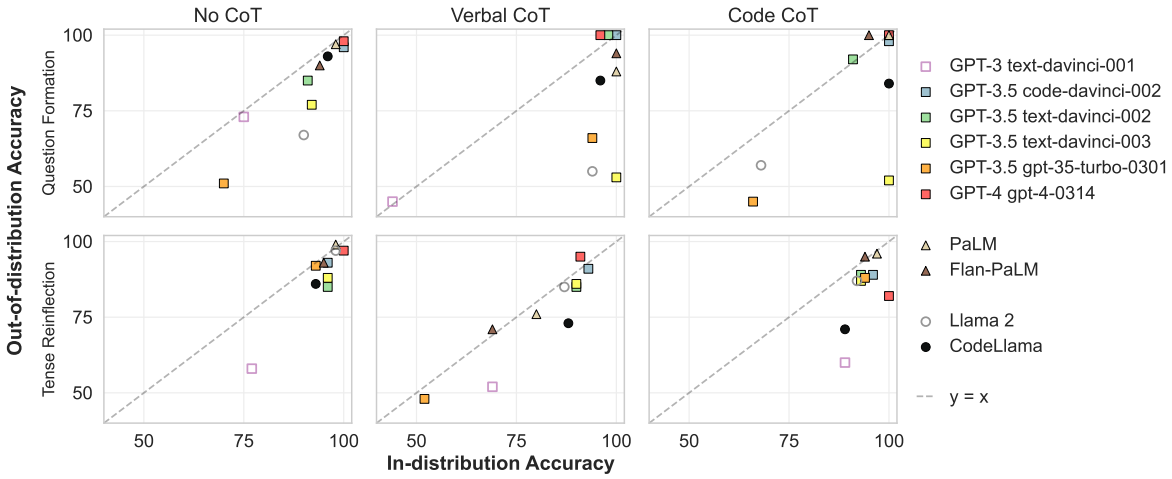


Figure 3: Main auxiliary accuracy on question formation and verb accuracy on tense reinflection. In-distribution accuracies reveal whether the models have learned the task, and out-of-distribution accuracies reveal whether models generalize robustly. Unfilled shapes (GPT-3, Llama 2) were trained on less than 5% code. We interpret the dashed line as a ceiling on OOD accuracy given ID accuracy.

wise describes which words correspond to which syntactic components, but in a Python code format where each component is represented as a variable, and then the model is explicitly instructed (in code) how to combine these components to form the final answer. The precise format of the inputs is shown in Figure 2. For the Code CoT prompt, the 8 exemplars exceed the maximum sequence length of GPT-3 and (Flan-)PaLM; in such cases, we use 4 Code CoT exemplars, which is the maximum that will fit while still allowing the model to generate the full reasoning trace and answer.[7]

2.4 Evaluation

We evaluate on a uniform subsample of 100 examples from the question formation generalization set of Mueller et al. (2022) and 100 examples from the tense reinflection generalization set of McCoy et al. (2020). To verify whether models have learned the task, we also present scores on 100-example uniform subsamples from the *in-distribution* test sets, where each example can be correctly transformed using either syntactic *or* positional features. All examples are generated from a probabilistic context-free grammar; as such, the distribution of syntactic structures is highly constrained, and the variation across examples is primarily lexical. We therefore reason that 100 examples should provide a representative sample of the syntactic structures in the evaluation sets.

To evaluate question formation examples, we use **main auxiliary accuracy**, which measures whether the model has moved the correct main auxiliary to the front of the sentence. This is measured by observing whether the first word is correct. For tense reinflection, we use **verb accuracy**, which measures whether each generated verb is correctly inflected *and* in the correct relative position.

3 Results

All models except GPT-3 and GPT-3.5 (Turbo) learn to perform the transformations tasks, as indicated by high in-distribution accuracies. Among models that learn the task well, there is large variance with respect to out-of-distribution accuracies.

Scale does not fully explain performance. GPT-3, GPT-3.5 `text-davinci-002` and `text-davinci-003` are estimated to be the same size, but the GPT-3.5 models generalize better. Llama 2 and CodeLlama outperform GPT-3 and (Flan-)PaLM despite being far smaller, and CodeLlama significantly outperforms Llama 2 on question formation despite being smaller. What explains differences between models, then?

Pre-training on code improves OOD generalization. On average, scores are higher given *any* prompt for models pre-trained on code: CodeLlama significantly outperforms Llama 2 on question formation (and performs comparably on tense reinflection), while GPT-3.5

[7] We observe in our results that (Flan-)PaLM ID and OOD performance are still high for Code CoT despite using only 4 exemplars, and that GPT-3 does not achieve high ID or OOD performance given *any* prompt (regardless of the number of exemplars).

code-davinci-002 outperforms all other GPT-3 and GPT-3.5 models. This agrees with and extends the finding of Mueller and Linzen (2023) that the domain of the pre-training corpus significantly affects syntactic generalization. This also agrees with recent findings that code pre-training improves other fundamental linguistic abilities (e.g., entity tracking; Kim and Schuster, 2023).

Conversely, **RLHF may harm OOD generalization.** GPT-3.5 text-davinci-003 performs at least as well as other GPT-3.5 models on in-distribution examples, but it generalizes consistently *worse* than other models which are *fine-tuned* on human demonstrations (including text-davinci-002 and CodeLlama). Thus, RLHF is effective when testing on in-distribution test sets, but it may actively harm a model's ability to reason beyond its given exemplars. This finding is preliminary, however, and should be confirmed in future work via ablations over RLHF in a variety of architectures and model sizes.

Chain-of-thought prompting has different impacts on in-distribution and out-of-distribution accuracies. For question formation, CoT prompting tends to increase ID accuracy while not improving OOD generalization; this is especially apparent for verbal CoT—and, for models trained on code, the code CoT prompt as well. This reveals the importance of testing OOD generalization when designing CoT prompts: **when chain-of-thought prompting improves performance on ID examples, improvements will not necessarily generalize to OOD examples.** For tense reinflection, verbal CoT equally harms both ID and OOD accuracy relative to No CoT prompts; conversely, Code CoT maintains ID accuracy while harming OOD accuracy. This generalizes the above point: depending on the task and prompt format, **CoT can have different impacts on ID and OOD performance**.

We qualitatively analyze model errors to investigate why models achieve imperfect OOD generalization (App. B). We find that code-davinci-002 is sensitive to variable names, and that most errors are because models move *affirmative* verbs instead of syntactically correct verbs.

Do the trends we have found in transformations tasks generalize to classification tasks? In App. D, we present a preliminary investigation into LLM performance on the HANS natural language inference benchmark (McCoy et al., 2019), with and without CoT prompting. We find that each model we test is susceptible to syntactic heuristics—including those pre-trained on code—and CoT often *increases* reliance on heuristics.

3.1 Do LLMs Generalize Faithfully?

LLMs have been found to generate answers which are not **faithful** to their own chain-of-thought reasoning (Lyu et al., 2023; Turpin et al., 2023). In this section, we test the relationship between models' CoT reasoning and their answers. There are at least two possibilities that could lead to low accuracy: First, the models could produce incorrect CoT reasoning traces, but answer consistently with those traces. Second, they could produce correct reasoning traces, but ignore those traces when producing the output. Here, we evaluate the accuracy of models' reasoning, as well as the faithfulness of models' final answers to their reasoning traces.

3.1.1 Method

The templatic format of the Code CoT prompt allows us to easily extract each component of the model's reasoning from the variable values. We observe whether the reasoning matches what the ground-truth values should be (**reasoning accuracy**) and whether the model's prediction aligns with the generated reasoning (**faithfulness**).

For question formation, we measure how often the model extracts the main subject, the main verb phrase (VP), and the object, and how often its output follows its analysis of each component:

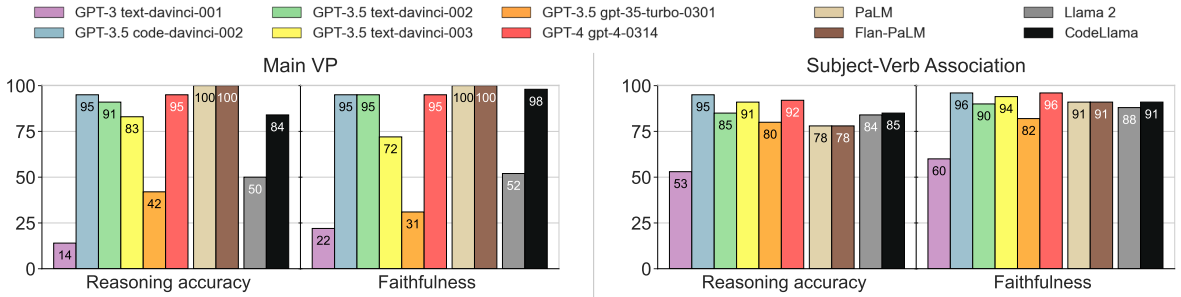
(6) The salamanders near my yaks that haven't entertained your unicorn have amused the newt. ⇒ Have the salamanders near my yaks that haven't entertained your unicorn amused the newt?

For tense reinflection reasoning, we measure whether each noun is detected and whether each verb is correctly associated with its subject (subject-verb association). When evaluating faithfulness, we also measure whether the verbs in the output agree with the subjects they were associated with in the CoT reasoning (verb number). For example, given the following prompt and answer:

(7) Your peacocks that admired the raven remembered your vulture. ⇒ Your peacocks that admire the raven remember your vulture.[8]

We present a subset of our analysis, focusing

[8] Alternating colors mean that a word is used in evaluating multiple components.



(a) Question formation

(b) Tense reinflection

Figure 4: Reasoning accuracies and faithfulness scores for question formation (left) and tense reinflection (right) using the Code CoT prompt. Reasoning accuracies and faithfulness are highest for `code-davinci-002`, GPT-4, (Flan-)PaLM, and CodeLlama.

on components that are essential for achieving the correct answer. See App. C for the full analysis and details on how we extract each component from a model's generated reasoning and answer.

3.1.2 Results

Reasoning accuracies and faithfulness (Figure 4) are highest for `code-davinci-002`, GPT-4, (Flan-)PaLM, and CodeLlama. These are the same models which had the highest OOD accuracies; indeed, OOD accuracies correlate strongly with reasoning accuracies ($\rho_{\text{Spearman}} = 0.71$, $p < .01$) and faithfulness scores ($\rho_{\text{Spearman}} = 0.71$, $p < .01$). This is unsurprising, but it still aids our understanding of *why* certain models generalize better. For question formation, models pre-trained on large amounts of code—notably, GPT-3.5 `code-davinci-002`, `text-davinci-002`, GPT-4, Flan-(PaLM), and CodeLlama—show high reasoning accuracies and faithfulness. GPT-3.5 `text-davinci-003` reasons less accurately, perhaps due to RLHF. Trends are similar for tense reinflection, except that Llama 2 and CodeLlama perform more similarly to each other and PaLM/Flan-PaLM perform worse than GPT-3.5 models. These results raise two hypotheses: (1) code pre-training induces better OOD generalization because it induces better (more accurate and faithful) reasoning, whereas (2) RLHF induces worse generalization because it optimizes features that are orthogonal to linguistic reasoning. These hypotheses are discussed in §4.

We also observe that **reasoning accuracy correlates strongly with faithfulness** ($\rho_{\text{Spearman}} = 0.87$, $p < .01$). Our hypothesis was that low performance could be explained by models either reasoning well but ignoring their reasoning, or reasoning poorly and answering accordingly. Instead, we find that a model's ability to follow its own reasoning is linked to how well it reasons. Future work should investigate to what extent accuracy, reasoning accuracy, and faithfulness are *causally* interdependent.

4 Discussion

When fine-tuned on transformations tasks, even small pre-trained LMs consistently achieve near-perfect accuracies when generalizing out of distribution (Mueller et al., 2022; Mueller and Linzen, 2023). In contrast, LLMs guided via in-context examples often rely on spurious positional and word-level features, as indicated by sub-perfect OOD accuracies and reliance on non-syntactic heuristics (like the MOVE-AFFIRMATIVE heuristic described in App. B). It is surprising that we often obtain outputs that do not align with the grammar of English, given that the training objective of language models directly optimizes for the production of probable sequences. Larger and deeper models generally behave in a manner more consistent with syntactic structure (Mueller and Linzen, 2023), but this trend does not generalize to ICL. Our findings provide evidence that, despite impressive performance on downstream tasks, **LLMs do not consistently leverage the latent structure of language when processing or generating language.**; rather, they detect and rely on a series of heuristics. LLMs will therefore struggle to generalize well outside of their exemplars' distribution when guided via ICL.

Code pre-training improves LLM generalization. Various studies observe that LMs trained on large amounts of code perform better on various NLP tasks and linguistic evaluations (Kim and Schuster, 2023; Madaan et al., 2022; Sap et al., 2022). Why is code such an effective signal when mixed with natural language? Some have speculated that code provides additional grounding

(Potts, 2020; Merrill et al., 2021, *inter alia*). Perhaps more importantly, code contains frequent instances of long-range state tracking, as well as hierarchically structured classes and function stacks; these may impart inductive biases that are helpful for learning hierarchical linguistic structure.

It is unclear why GPT-3.5 `text-davinci-003`—the only model in the GPT-3.5 family trained with RLHF—performs significantly worse than comparable GPT-3.5 models. This goes against conventional wisdom that RLHF generally improves performance on downstream NLP tasks (Ye et al., 2023),[9] but corroborates preliminary evidence that RLHF degrades certain aspects of performance, like certainty calibration.[10] Further work is needed to fully understand RLHF's impact on generalization and whether these findings are causally linked. Perhaps the reinforcement learning procedure optimizes models to generate language that *semantically* aligns with human quality judgments, but at the cost of causing the model to assign lower importance to syntactic features. Meanwhile, providing human feedback via fine-tuning may be less optimal for aligning outputs to human judgments, but better for preserving sensitivity to syntax.

Larger models often perform better on many NLP tasks—at least where there is no large distribution shift. However, our findings reveal that **scale is not a panacea for robust reasoning and language understanding**: rather, other factors like training objectives, the type of pre-training data, and the supervision method(s) make a significant difference. This agrees with and extends findings from Mueller and Linzen (2023) that the domain of the pre-training data significantly influences how models generalize.

5 Related Work

Since the discovery that LMs are capable of ICL, studies have explored ICL's limits (e.g., Akyürek et al. 2023; Chan et al. 2022). Analyses of ICL have revealed counterintuitive biases: for example, correct labels are not necessary for strong performance (Min et al., 2022b), and models can perform very well even given misleading/irrelevant prompts (Webson and Pavlick, 2022) or flipped/semantically misleading label names (Wei et al., 2023; Pan et al., 2023). In contemporaneous work, Si et al. (2023a) analyze which semantic features (e.g., sentiment vs. topic) LLMs prefer by designing underspecified exemplars. Their method is similar to ours in that they use ambiguous exemplars and disambiguating test examples; however, we apply this approach to *syntactic* processes to analyze the fundamental linguistic structural abilities of LLMs.

Any finite training set is consistent with multiple generalizations. Studies have observed that LSTM- and Transformer-based encoder-decoder models trained from scratch on syntactic transformations *do not* generalize in a syntax-sensitive manner (McCoy et al., 2018, 2020; Petty and Frank, 2021), but pre-trained encoder-decoder models *do* generalize syntactically (Mueller et al., 2022). Similar positive results have been reported for RoBERTa, but only after large-scale pretraining (Warstadt and Bowman, 2020; Warstadt et al., 2020).

Most similarly to our work, Hu and Levy (2023) prompt LLMs on metalinguistic judgments; they find that prompting underestimates syntactic awareness compared to probability measurements. Our findings extend this conclusion: this issue may be even more pronounced when test examples are not identically structured to the exemplars given to the model. When fine-tuning, larger models generally have inductive biases that align more strongly with syntactic structure (Mueller and Linzen, 2023), and so it is plausible to predict that greater scale should also lead to more syntax-sensitive behavior when using ICL. Nonetheless, this is not what we find.

Crucially, most benchmarking tasks rely on data where the training and test sets are drawn from the same distribution, which limits our understanding of how well models truly generalize (Linzen, 2020). Saparov and He (2023) find that LLMs are prone to relying on spurious correlations that hinder robust generalization. Drozdov et al. (2023) find that a series of chain-of-thought prompts can yield more robust generalization, however.

6 Conclusions

We have investigated how well LLMs generalize out-of-distribution on tasks requiring syntax for robust performance. Our findings reveal significant variance across models that is not fully explained by scale. Models trained on code are better at leveraging in-context examples to generalize more robustly, and at reasoning accurately and faithfully—even at smaller scales.

[9] OpenAI also officially states that `text-davinci-003` is more capable than `text-davinci-002` in multiple locations on their website, including here and here.

[10] `https://openai.com/research/gpt-4`

Limitations

Using closed-source language models presents many scientific challenges. There are no public resources that contain exact information about the models we test, such as their parameter counts, training distributions, and corpus sizes, among other important details. We also cannot definitively rule out that the transformations data is contained in the pretraining corpus. When asking the GPT models directly if they have seen the dataset or its GitHub repository, they did not state that they have seen them. However, Bard (not used in this study) directly admits to having seen syntax-aware evaluation datasets like HANS.

OpenAI support for the Codex model was discontinued while this project was in progress. This reveals another set of challenges of working with closed-sourced models in scientific contexts: replicability becomes difficult, and access is dependent on the support of non-accountable entities. While we were able to attain access again by applying to a research program, access is not guaranteed indefinitely.

Regarding our experimental design, it is difficult to disentangle whether model errors are due to subhuman syntactic reasoning abilities, faulty inductive biases, or simply poor ICL abilities (App. B). We partially controlled for this by evaluating on in-distribution transformations, which evaluates how well models can leverage ICL on in-distribution examples. However, the transformations task does not in itself tell us what specific *mechanism* leads to incorrect results: perhaps LLMs do robustly represent the syntactic structure of input sentences, but preferentially rely on superficial features. Or, perhaps they do not robustly represent sentence structure, instead relying on a series of heuristics that allow certain models to approximate syntactic generalization in specific circumstances. Future work should employ mechanistic interpretability methods to uncover what, precisely, causes errors in transformations tasks.

Ethics Statement

Regardless of how well large language models generalize or to what extent they demonstrate language understanding, they are still eminently capable of misuse and harm. LLMs excel at generating convincing arguments for false conclusions, or for generating false information and confidently presenting it as fact—for example, when generating fake news. Our findings may erroneously suggest to some readers that addressing these concerns can wait, as LMs are still incapable of robustly understanding sentence structure; however, we disagree with this takeaway. Research that addresses the harms of LLMs should occur in parallel with research that investigates (and improves) their decision-making mechanisms.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

Edwin L Battistella. 1996. *The Logic of Markedness*. Oxford University Press.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Stephanie C. Y. Chan, Ishita Dasgupta, Junkyung Kim, Dharshan Kumaran, Andrew K. Lampinen, and Felix Hill. 2022. Transformers generalize differently from information stored in context vs in weights. *Computing Research Repository*, arXiv:2210.05675.

Angelica Chen, Ravid Schwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. 2023. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in mlms. *Computing Research Repository*, arXiv:2309.07311.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan

Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Computing Research Repository*, arXiv:2107.03374.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *Computing Research Repository*, arXiv:2204.02311.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Computing Research Repository*, arXiv:2210.11416.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognizing textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*.

Robert Frank and Donald Mathis. 2007. Transformational networks. In *Proceedings of the Workshop on Psychocomputational Models of Human Language Acquisition*. Cognitive Science Society.

Joseph H Greenberg. 1966. *Language Universals: With Special Reference to Feature Hierarchies*. De Gruyter Mouton.

Jennifer Hu and Roger Levy. 2023. Prompting is not a substitute for probability measurements in large language models. *Computing Research Repository*, arXiv:2305.13264.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

Henry Kucera. 1982. Markedness and frequency: A computational analysis. In *Coling 1982: Proceedings of the Ninth International Conference on Computational Linguistics*.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *Computing Research Repository*, arXiv:2301.13379.

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

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: Hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, pages 2096–2101, Madison, Wisconsin. Cognitive Science Society.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does Syntax Need to Grow on Trees? Sources of Hierarchical Inductive Bias in Sequence-to-Sequence Networks. *Transactions of the Association for Computational Linguistics*, 8:125–140.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

William Merrill, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. 2021. Provable Limitations of Acquiring Meaning from Ungrounded Form: What Will Future Language Models Understand? *Transactions of the Association for Computational Linguistics*, 9:1047–1060.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022a. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022b. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Aaron Mueller, Robert Frank, Tal Linzen, Luheng Wang, and Sebastian Schuster. 2022. Coloring the blank slate: Pre-training imparts a hierarchical inductive bias to sequence-to-sequence models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1352–1368, Dublin, Ireland. Association for Computational Linguistics.

Aaron Mueller and Tal Linzen. 2023. How to plant trees in language models: Data and architectural effects on the acquisition of syntactic inductive biases. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Toronto, ON. Association for Computational Linguistics.

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

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics (ACL)*.

Jackson Petty and Robert Frank. 2021. Transformers generalize linearly. *Computing Research Repository*, arXiv:2109.12036.

Christopher Potts. 2020. Is it possible for language models to achieve language understanding? Medium blog post.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *Computing Research Repository*, arXiv:2308.12950.

Maarten Sap, Ronan Le Bras, Daniel Fried, and Yejin Choi. 2022. Neural theory-of-mind? on the limits of social intelligence in large LMs. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3762–3780, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations*.

Chenglei Si, Dan Friedman, Nitish Joshi, Shi Feng, Danqi Chen, and He He. 2023a. Measuring inductive biases of in-context learning with underspecified demonstrations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Toronto, ON. Association for Computational Linguistics.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2023b. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Computing Research Repository*, arXiv:2307.09288.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Warstadt and Samuel R. Bowman. 2020. Can neural networks acquire a structural bias from raw

linguistic data? In *Proceedings of the 42nd Annual Meeting of the Cognitive Science Society*, Online. Cognitive Science Society.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel R. Bowman. 2020. Learning which features matter: RoBERTa acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023. Larger language models do in-context learning differently. *Computing Research Repository*, arXiv:2303.03846.

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

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, et al. 2023. A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. *Computing Research Repository*, arXiv:2303.10420.

A Tense Reinflection Prompt Formats

In Figure 5, we present the prompt formats we use for the tense reinflection task.

B Ablations and Error Analysis

In this appendix, we analyze which aspects of the prompt affect OOD generalization and investigate the type of errors that LMs make.

When reasoning with code, GPT-3.5 `code-davinci-002` is sensitive to variable names, but GPT-4 is not. GPT-4 and `code-davinci-002` perform best on both transformations, and seem to improve with Code CoT prompting. We note that in the prompts we used in the main text, the variables had semantically meaningful names, such as `main_subject`; here, we ask how crucial this factor is for the models' ability to benefit from the Code CoT prompt. We present the prompt formats we use for this analysis in Figure 6.

We first replace variable names (like `main_auxiliary` and `main_subject`) with nonce words (like `wug` and `dax`). We ensure that the same variable name is used for the same syntactic component in each exemplar, such that the model can form meaningful associations with the nonce (or swapped, as in the following analysis) variable names. For `code-davinci-002`, this reduces 8-shot performance from 97% to 76%. For GPT-4, this only reduces 8-shot performance from 97% to 96%.

In a second experiment, we assign variable names adversarially, such that seemingly meaningful variable names are no longer associated with their conventional linguistic category; for example, the main verb phrase is associated with the variable `main_object` (e.g., `main_object = "has entertained"`), or the main subject is associated with the variable `main_vp` (e.g., `main_vp = "The newt"`). For `code-davinci-002`, this reduces performance from 98% to 83%, but for GPT-4, the accuracy stays the same at 97%. These patterns indicate that `code-davinci-002` is sensitive to—and may usably understand and rely on—terminology like "subject" and "verb". Meanwhile, GPT-4 may or may not understand the "subject" and "verb" names, but it does not crucially rely on them to perform the task; it instead shows an ability to adapt to arbitrary variable names.

No Chain of Thought

Input

Q: Convert this sentence to present tense. Some salamanders who smiled admired her newts.
A: Some salamanders who smile admire her newts.

… *(up to 7 more exemplars)*

Q: Convert this sentence to present tense. The yak upon my ravens entertained her zebras.
A:

Output

The yak upon my ravens entertains her zebras.

Verbal Chain of Thought

Input

Q: Convert this sentence to present tense. Some salamanders who smiled admired her newts.
A: "salamanders" is the main subject. "smiled" agrees with "salamanders". "admired" agrees with "salamanders". We need to convert every verb to present tense. "salamander" is singular, so smiled becomes smile. "salamander" is singular, so admired becomes admire. The answer is Some salamanders who smile admire her newts.

… *(up to 7 more exemplars)*

Q: Convert this sentence to present tense. The yak upon my ravens entertained her zebras.
A:

Output

"yak" is the main subject. "ravens" is inside a prepositional phrase. "entertained" agrees with "yak". We need to convert every verb to present tense. "yak" is singular, so entertained becomes entertains. The answer is The yak upon my ravens entertains her zebras.

Code Chain of Thought

Input

Q: Convert this sentence to present tense. Some salamanders who smiled admired her newts.
A:

```
# Write a Python script to extract each subject and their associated verbs.
# Using this information, transform each past-tense verb into a present-tense
verb by agreeing the verbs with their subjects.
sentence = "Some salamanders who smiled admired her newts."
answer = sentence

# keys are subjects, values are associated verbs
subjects_verbs = {"salamanders": ["smiled", "admired"], "newts": []}

for subject in subjects_verbs.keys():
  for verb in subjects_verbs[subject]:
    if is_singular(subject):
      present_verb = make_singular(verb)
    else:
      present_verb = make_plural(verb)
    answer.replace(verb, present_verb, 1)
# The answer is Some salamanders who smile admire her newts.
```

… *(up to 7 more exemplars)*

Q: Convert this sentence to present tense. The yak upon my ravens entertained her zebras.
A:

Output

```
# Write a Python script to extract each subject and their associated verbs.
# Using this information, transform each past-tense verb into a present-tense
verb by agreeing the verbs with their subjects.
sentence = "The yak upon my ravens entertained her zebras."
answer = sentence

# keys are subjects, values are associated verbs
subjects_verbs = {"yak": ["entertained"], "ravens": [], "zebras": []}

for subject in subjects_verbs.keys():
  for verb in subjects_verbs[subject]:
    if is_singular(subject):
      present_verb = make_singular(verb)
    else:
      present_verb = make_plural(verb)
    answer.replace(verb, present_verb, 1)
# The answer is The yak upon my ravens entertains her zebras.
```

Figure 5: The prompt formats we use for tense reinflection. We give the model up to 8 exemplars, followed by a test example. The model must generate the chain-of-thought reasoning *and* the answer. We highlight the answers with a blue background. In the code prompt, we put code (except comments) in `blue text`.

Code CoT: Nonce Variable Names

Input

Q: Transform this into a yes/no question. The yak has entertained our newts near the salamander.
A:

```
# Write a Python script to extract the main subject, main verb phrase, and
any distractor clauses/phrases from the text.
# Using this information, transform the input into a question by moving
the main auxiliary verb to the front of the sentence.
wug = "The yak"
dax = "our newts"
bouba = "has entertained"
kiki = "has"
blickets = ["near the salamander"]

words = sentence.split()
kiki_idx = words.index(kiki)
output_words = words.insert(0, words.pop(kiki_idx))
output_sentence = "".join(output_words).capitalize()
answer = output_sentence.replace(".", "?")
# The answer is Has the yak entertained our newts near the salamander?
```

… *(up to 7 more exemplars)*

Q: Transform this into a yes/no question. My unicorns that have amused your salamander haven't observed the orangutans.
A:

Output

```
# Write a Python script to extract the main subject, main verb phrase,
and any distractor clauses/phrases from the text.
# Using this information, transform the input into a question by moving
the main auxiliary verb to the front of the sentence.
wug = "My unicorns"
dax = "the orangutans"
bouba = "haven't observed"
kiki = "haven't"
blickets = ["that have amused your salamander"]

words = sentence.split()
kiki_idx = words.index(kiki)
output_words = words.insert(0, words.pop(kiki_idx))
output_sentence = "".join(output_words).capitalize()
answer = output_sentence.replace(".", "?")
# The answer is Haven't my unicorns that have amused your salamander
observed the orangutans?
```

Code CoT: Swapped Variable Names

Input

Q: Transform this into a yes/no question. The yak has entertained our newts near the salamander.
A:

```
# Write a Python script to extract the main subject, main verb phrase, and
any distractor clauses/phrases from the text.
# Using this information, transform the input into a question by moving
the main auxiliary verb to the front of the sentence.
main_vp = "The yak"
main_aux = "our newts"
main_object = "has entertained"
main_subject = "has"
distractors = ["near the salamander"]

words = sentence.split()
main_subject_idx = words.index(main_subject)
output_words = words.insert(0, words.pop(main_subject_idx))
output_sentence = "".join(output_words).capitalize()
answer = output_sentence.replace(".", "?")
# The answer is Has the yak entertained our newts near the salamander?
```

… *(up to 7 more exemplars)*

Q: Transform this into a yes/no question. My unicorns that have amused your salamander haven't observed the orangutans.
A:

Output

```
# Write a Python script to extract the main subject, main verb phrase,
and any distractor clauses/phrases from the text.
# Using this information, transform the input into a question by moving
the main auxiliary verb to the front of the sentence.
main_vp = "My unicorns"
main_aux = "the orangutans"
main_object = "haven't observed"
main_subject = "haven't"
distractors = ["that have amused your salamander"]

words = sentence.split()
main_subject_idx = words.index(main_subject)
output_words = words.insert(0, words.pop(main_subject_idx))
output_sentence = "".join(output_words).capitalize()
answer = output_sentence.replace(".", "?")
# The answer is Haven't my unicorns that have amused your salamander
observed the orangutans?
```

Figure 6: Prompt formats for our Code CoT analyses. For each exemplar, we ensure that the same variable name is used for the same syntactic component, such that the model can form meaningful associations with the nonce/swapped variable names.

GPT-3 and GPT-3.5 (Turbo) rely heavily on spurious lexical features. What kinds of errors are causing such low performance in GPT-3 and GPT-3.5 (Turbo)? In a qualitative analysis, we find that when these models do not produce the correct answer, it is typically because they move the *affirmative* (non-negated) auxiliary, regardless of whether it is the main auxiliary. We call this the MOVE-AFFIRMATIVE heuristic. For example, in Ex. (8-a) and (8-b) below, these models would likely move the affirmative auxiliary, even though it results in the incorrect output in Ex. (8-b).

(8) a. The unicorn that **hasn't** entertained the newts **has** observed the yak. ⇒ **Has** the unicorn that **hasn't** entertained the newts observed the yak? ✓

b. The unicorn that **has** entertained the newts **hasn't** observed the yak. ⇒ **Has** the unicorn that **has** entertained the newts observed the yak? ✗

In fact, for *all* models we test, errors in question formation are typically due to reliance on MOVE-AFFIRMATIVE. The fact that models adopt this heuristic is puzzling, as it is not consistent with the in-context exemplars (half of the verbs in the exemplars are negative verbs), and therefore reflects an inductive bias. We hypothesize that the bias in question favors generating linguistically unmarked forms in ambiguous contexts. Such an unmarkedness bias would be easy to learn given that unmarked forms are generally more frequent (Greenberg, 1966). In this case, affirmative verb forms are likely more frequent than negative ones, especially at the beginning of a question. [11] More broadly, a systematic preference for linguistically unmarked forms could be a form of overfitting to the training data. Future work could assess whether this type of overfitting is more likely with scale, or whether it could be overcome with some form of intervention to the data or model.

In tense reinflection, most errors are due to models relying on the linear AGREE-RECENT hypothesis. Thus, errors in this case reflect reliance on word position and relative word ordering in generating sentences. All models we test generalize in a syntax-sensitive manner more often than not, but this error pattern reflects that models are not *entirely* dependent on syntactic inductive biases: rather, given in-context examples, models can generalize using either inductive bias.

[11] The relationship between markedness and frequency is disputed (Battistella, 1996). Frequency and markedness often correlate, but not always; see Kucera (1982) for an empirical case study in English.

C Faithfulness of Chain-of-Thought Reasoning Traces: Full Analysis

C.1 Extracting Components from Model Reasoning

Question formation. The model must generate variables corresponding to the main subject, main object, and main verb phrase. For example, given Ex. (6), the model must predict `main_subject = "The salamanders"` for its reasoning to count as correctly identifying the main subject. For faithfulness, if the model predicts `main_subject = "The salamanders"` in its reasoning and then begins its answer with "Have the salamanders…", this would be considered faithful to its own reasoning on the main subject; conversely, if it predicts `main_subject = "The salamanders"` in its reasoning but then begins its final answer with "Have the yaks…", this would not be considered faithful.

Tense reinflection. The model must generate a Python dictionary of subjects (keys) and verbs (values) as part of its CoT prompt. In this example, the correct dictionary is `subjects_verbs = {"peacocks": ["admired", "remembered"], "raven": [], "vulture": []}`. The noun reasoning accuracy is the proportion of nouns from the sentence that are present in the dictionary, while the subject-verb association accuracy is the proportion of verbs associated with the correct subject key. For faithfulness, we measure how many nouns from the dictionary are present in the answer, and whether the verbs' subjects in the answer are the same as they are in the dictionary (i.e., whether the sentence structure is compatible with the reasoning). For verb number faithfulness, we measure the proportion of verbs in the dictionary whose grammatical number in the final answer is the same as the grammatical number of their subject in the dictionary.

C.2 Full Results

Here, we present reasoning accuracies and faithfulness scores for all reasoning components, models, and tasks. See Figure 7 for results on question formation, and Figure 8 for results on tense reinflection.

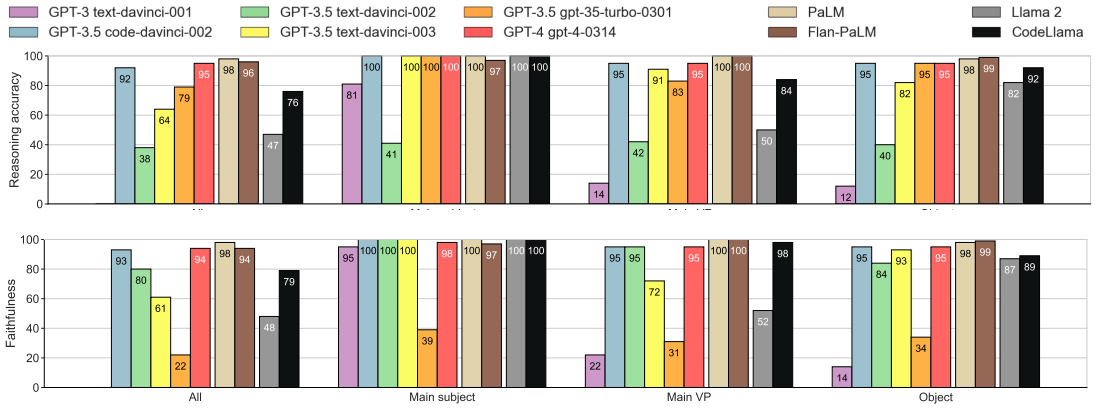


Figure 7: Reasoning accuracies and faithfulness scores for various models and syntactic components using the Code CoT prompt on question formation. "All" refers to the proportion of examples where models correctly label each syntactic component, or where outputs are entirely faithful to the model's generated reasoning.

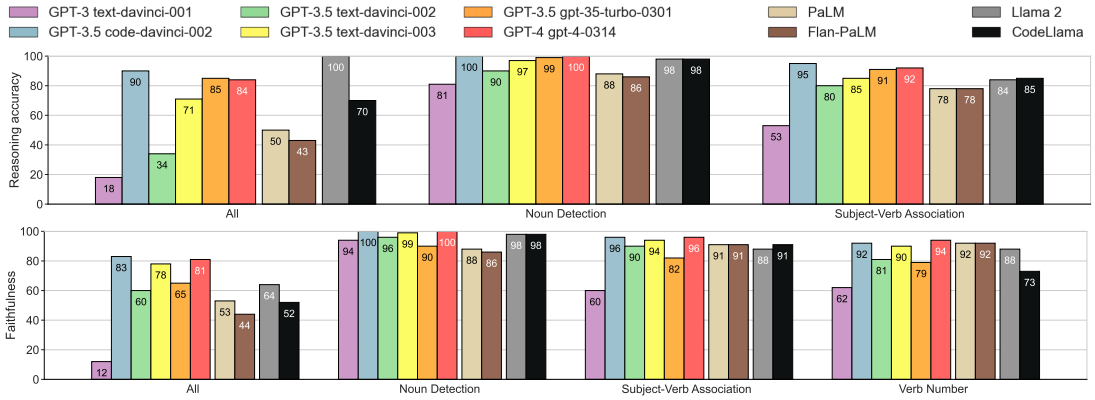


Figure 8: Reasoning accuracies and faithfulness scores for various models and syntactic components using the Code CoT prompt on tense reinflection. "All" refers to the proportion of examples where models correctly label each syntactic component, or where outputs are entirely faithful to the model's generated reasoning.

D Syntactic Generalization in Text Classification: The Case of Natural Language Inference

The analyses reported in the main text found that LLMs leverage syntactic information to varying extents when taught a task via ICL, and that code pre-training improves OOD generalization. In this section, we test if these trends are specific to tasks where models need to generate transformed versions of their inputs, or if they generalize to classification tasks. We leverage the Heuristic Analysis for NLI Systems (HANS; McCoy et al., 2019) dataset, which consists of Natural Language Inference (NLI) examples designed to diagnose reliance on syntactic heuristics. One typically trains the model on the training set of another NLI dataset, such as MNLI (Williams et al., 2018) or RTE (Dagan et al., 2006), and then uses HANS as an evaluation set to measure whether models rely on syntactic heuristics. Si et al. (2023b) evaluate RoBERTa and GPT-3 on HANS after training on MNLI; they find that both models perform similarly, suggesting that scale may not be enough for models to overcome syntactic heuristics. We extend this analysis to a wider variety of LLMs and to chain-of-thought prompting, and further break down performance by label.

HANS disagnoses reliance on three particular heuristics: lexical overlap, subsequence, and constituent heuristics. A model relying on **lexical overlap** heuristics assumes that two sentences with high word overlap entail each other. For example, although "The actor paid the doctor" means the opposite of "The actor was paid by the doctor", a model could assume these are entailed because of the high amount of word overlap between the two sentences. A model relying on **subsequence** heuristics assumes that a hypothesis is entailed if it is a subsequence in the premise. For example, although "The actor near the doctor danced" does not imply that "The doctor danced", a model may assume

Model	Lexical Overlap		Subsequence		Constituent	
	No CoT	Verbal CoT	No CoT	Verbal CoT	No CoT	Verbal CoT
GPT-3 `text-davinci-001`	59	100	55	97	59	99
GPT-3.5 `code-davinci-002`	81	99	99	96	96	96
GPT-3.5 `text-davinci-002`	84	98	67	99	64	95
GPT-3.5 `text-davinci-003`	96	99	100	100	96	100
GPT-3.5 `gpt-3.5-turbo-0301`	81	16	69	3	96	6
GPT-4 `gpt-4-0314`	94	98	100	96	98	98
PaLM	94	98	99	100	100	100
Flan-PaLM	99	100	100	100	100	100
Llama 2	84	93	98	100	99	100
CodeLlama	80	65	95	50	91	89

(a) Entailment

Model	Lexical Overlap		Subsequence		Constituent	
	No CoT	Verbal CoT	No CoT	Verbal CoT	No CoT	Verbal CoT
GPT-3 `text-davinci-001`	86	0	76	0	85	6
GPT-3.5 `code-davinci-002`	99	94	70	48	74	35
GPT-3.5 `text-davinci-002`	98	84	86	48	88	38
GPT-3.5 `text-davinci-003`	99	99	50	34	61	11
GPT-3.5 `gpt-3.5-turbo-0301`	82	5	27	11	9	12
GPT-4 `gpt-4-0314`	94	92	48	46	49	35
PaLM	97	94	57	57	27	13
Flan-PaLM	78	96	36	41	3	2
Llama 2	99	91	78	51	27	11
CodeLlama	96	100	56	89	55	47

(b) Non-entailment

Table 2: Accuracies on HANS. Scores are split by label: test examples whose label is "entailment" are presented in the top table (a), while examples whose label is "non-entailment" are presented in the bottom table (b). High scores on entailment coupled with low scores on non-entailment signify that the model relies on the syntactic heuristic to predict the label.

they are entailed because the latter sentence is fully contained in the first. Finally, a model relying on **constituent** heuristics assumes that a hypothesis is entailed if it is a subtree of the premise. For example, "If the artist slept, the doctor ran" does not imply that "The artist slept", but a model relying on this heuristic would assume that it is entailed. Models that systematically rely on any of these heuristics are expected to obtain 100% accuracy on examples where the label is "entailment" because the heuristics happen to make the correct prediction; however, models that rely on these heuristics will also obtain 0% accuracy on examples where the label is "non-entailment".

We evaluate our LLMs on HANS in the ICL setting. The 8 ICL exemplars are drawn from MNLI, and the test examples are from HANS. Our results are based on a uniform subsample of 100 examples per heuristic *and* per label. We test across 3 syntactic heuristics, and there are 2 labels (entailment and non-entailment); thus, we have 600 test examples in total. We use No CoT prompts and Verbal CoT prompts similar to those we depict in Figure 2. For example:

(9) **No CoT**

a. *Input:*
Q: There are many regulations in place that will reduce air emissions from electric power generation. Is it definitely true that There are a lot of regulations in place that reduce emissions?
A: Yes

...(up to 7 more exemplars from MNLI)

Q: The lawyer was advised by the actor. Is it definitely true that The lawyer advised the actor?
A:

b. *Output:*
Yes

(10) **Verbal CoT**

a. *Input:*
Q: There are many regulations in place that will reduce air emissions from electric power generation. Is it definitely true that There are a lot of regulations in place that reduce emissions?
A: Regulations is the agent of both sentences. Emissions is the object inside the clauses in both sentences. Reduce is inside a clause in both sentences. The answer is Yes.

...(up to 7 more exemplars from MNLI)

Q: The lawyer was advised by the actor. Is it definitely true that The lawyer advised the actor?
A:

b. *Output:*
The lawyer is the subject of the first sentence. The actor is the subject of the second sentence. Advised is the verb in the first sentence. Was advised is the verb in the second sentence. The answer is Yes.

Our results (Table 2) suggest that LLMs are susceptible to syntactic heuristics, but to a lesser extent than smaller-scale LMs fine-tuned on MNLI (McCoy et al., 2019). Most models achieve close to 100% accuracy on examples where the gold label is "entailment", as expected. However, on "non-entailment" examples, variance is very high: all models (except GPT-3.5 (Turbo)) achieve high accuracies on lexical overlap examples using both prompts, suggesting that they do *not* succumb to the lexical overlap heuristic when generalizing. For other syntactic heuristics, however, non-entailment scores tend toward 0–50%. This suggests that models rely on subsequence and constituent heuristics when guided via ICL, though the extent to which they rely on the heuristic depends on the prompt format. Unlike in the transformation tasks, here code pre-training does not seem to lend a significant advantage: there is no consistent significant gain (nor loss) on entailment or non-entailment examples in models pre-trained on code.

We also find that **chain-of-thought reasoning makes models more prone to relying on syntactic heuristics.** With Verbal CoT, scores on entailment examples tend to increase across models and across syntactic heuristic types. On non-entailment examples, however, it tends to reduce scores.