Pelican Soup Framework: A Theoretical Framework for Language Model Capabilities

Anonymous ACL submission

Abstract

In this work, we aim to better understand how pretraining allows LLMs to (1) generalize to unseen instructions and (2) perform in-context learning, even when the verbalizers are irrelevant to the task. To this end, we propose a simple theoretical framework, Pelican Soup, basing on the logical soundness of the training data, a notion of "reference-sense association" and a simple formalism for natural language processing tasks. Our framework demonstrates how linguistic, psychology, and philosophy studies can inform our understanding of the language model and is connected to several other existing theoretical results. As an illustration of the usage of our framework, we derive a bound on in-context learning loss with our framework. Finally, we support our framework with empirical experiments and provide possible future research directions.

1 Introduction

Large language models (LLMs) have demonstrated the capability to perform downstream natural language processing (NLP) tasks. By following instructions, LLMs can perform tasks with zero-shot examples, demonstrating its reasoning capability. With some input-output examples provided in the prompt, LLMs can also perform tasks without instructions, which is known as in-context learning (ICL) (Chowdhery et al., 2022). Particularly, Brown et al. (2020) show that LLMs can perform ICL for classification tasks even when the verbalizers (labels present in the demonstration) are semantically irrelevant to the task, e.g., foo/bar instead of negative/positive (Wei et al., 2023). However, it is unclear how pretraining with a large amount of data leads to these capabilities.

To explain how LLMs acquire these capabilities, we propose a simple theoretical framework, the Pelican Soup framework in §2. Our framework is based on some very general assumptions, such as the logical soundness of the paragraphs in the training set and the freedom of expression-sense association in language (as discussed in the theory of meaning by philosopher Frege (1948, 1879)). Our framework also include a simple formalism for NLP tasks, which help explains why LMs can follow instructions.

In §3, we showcase how we can use this framework to analyze LLMs' ICL capability, which mitigates limitations of previous theoretical analyses. For example, in the first theoretical analyses of ICL, Xie et al. (2022) assumes that the general text for training LMs is from a hidden Markov model (HMM), which may be an oversimplification of natural language. In comparison, our framework does not require this strong assumption.

Our framework also makes the analysis more insightful than the one presented by Zhang et al. (2023). While the generation process by Zhang et al. (2023) is more general than the HMM assumption by Xie et al. (2022), it lakes groundings to real-world linguist phenomena. Our framework mitigates this limitation. It helps us better explain the physical meaning of the terms in the bound on ICL loss and show how the terms in our reflect real-world practices, such as instruction-tuning, the choice of verbalizers, and the distribution of prompts.

Furthermore, in §4, inspired by the cognitive science theories Fodor (1975, 2008); Piantadosi (2021), early development of artificial intelligence (AI) (Siskind, 1996; Murphy, 2004) and formal linguists Carnap et al. (1968); Bresnan and Bresnan (1982); Steedman (1987, 1996); Sag et al. (1999), we provide an extension of our framework to explain why generalization is possible. The extension also connects our framework to other theoretical results. For example, our extension instantiates the *complex skills* in the theory by (Arora and Goyal, 2023). In §5, with an extra assumption, the extension allows us to achieve a similar result as the one by Hahn and Goyal (2023), which bounds the ICL loss with the description length of the underlying input-label mapping function.

Our work informs future LLMs research directions. Scientifically, we shed light on how linguistic phenomena allow LLMs to acquire the surprising capabilities they exhibit during inference time. The framework also shows how linguistic, psychology and philosophy studies can inform our understanding of modern NLP. Practically, we highlight the importance of acquiring knowledge about the interrelation between concepts through pretraining. As shown in previous studies, the language modeling objective is inefficient for knowledge acquisition (Allen-Zhu and Li, 2023; Chiang et al., 2024). We suggest developing a better pretraining technique is crucial for future NLP development. Furthermore, we proposed experiment setups that mimic the acquisition of ICL and instruction-following capability on a tiny scale. These setups will facilitate future studies for better insights.

2 The Pelican Soup Framework

We aim our theoretical framework at explaining why LLMs can perform well on prompts for downstream tasks even though the prompts have a different distribution than the training corpus. Therefore, our framework includes assumptions qualifying the training corpus distribution in (§2.1) and a formalism for NLP tasks (§2.2). Later, we will show how this framework allows us to bound the loss of ICL.

2.1 Training Data Distribution

Our theory framework is based on the interrelations between the semantics of sentences. Thus, we first make a general assumption:

Assumption 2.1 (Sentence)**.** We assume that a sentence in a language is a sequence of words such that humans can determine whether one sentence entails or contradict another sentence.

Assumption 2.1 allows us to specify the combinations of sentences that can co-occur in a paragraph with non-zero possibility:

Assumption 2.2 (Soundness)**.** Any paragraph with non-zero probability mass is a set of sentences such that for any two sentences x_1, x_2 in the paragraph, x_1 does not contradict with x_2.

To show how modeling natural language leads to the ICL capability, we further introduce the notion of expression-sense association as a latent variable. It reflects the fact that language allows us to associate senses with expressions quite freely, as discussed in the theory of meaning As discussed in the theory of meaning (Frege, 1948, 1879), language allows us to associate senses with expressions quite freely. For example, when "she" or the human name "Emily" is present in a paragraph, it is associated with a certain person of certain characteristics, which reflect its sense.

Meanwhile, the usage of the expression is dependent on the sense it is associated with and is consistent within its context. For example, if "she" is associated with the sentence "a person who has a house", then by Assumption 2.2, the sentence "she has no property" will have 0 probability mass. Moreover, when we want to refer to "the person who has a house" instead of repeating the sentence again, we use "she" as an abbreviation.

For simplicity, we only consider single-word expressions and assume that such association is consistent throughout a document, and we assume the sense of an expression can be described with a set of sentences:

Assumption 2.3 (Expression-sense association)**.** There is a set of words Γ such that for every document in the training data, some $r \in \Gamma$ in the document is associated with a sense represented as a set of sentences Z_r with a prior distribution $\Pr(Z_r)$. Any $z \in Z_r$, z present in the document can be replaced with r without breaking the logical soundness of the document.

Adjectives such as "good" and "bad" are expressions that can be associated with variable senses too, and their sense also depends on the context. However, their meaning may not be as variable as pronouns. We reflect this difference with a prior distribution for the sense an expression is associated with in our theoretical analysis later.

Finally, we assume a document is a set of paragraphs where some expressions in Γ are present:

Assumption 2.4 (Document)**.** A document is a concatenation of paragraphs containing $r \in \Gamma$ separated with a delimiter d (e.g., a blank line).

2.2 A Formalism for NLP Tasks

With Assumption 2.1, we propose a simple formalism: For any objective or prescriptive NLP task (Rottger et al., 2022) that maps an input x to an output y[1], that task can be described with some

[1] We can generalize y to be a set to account for one-to-many generation tasks.

task instructions u such that $u \wedge x \models y$.

When it is a classification task, y is a label with a description v_y specified in the instruction u. x belongs to class y if and only if based on the instruction u, x entails v_y. Note that because we can rewrite $u \wedge x \models y$ as $x \models u \rightarrow v_y$, we can see $u \rightarrow v_y$ as the label description and absorb the symbol u as part of v_y. Thus, we can represent classification tasks with the class descriptions.

For example, we can formulate the sentiment analysis task over movie reviews as $\langle v_+, v_- \rangle =$ ⟨"I like the movie", "I dislike the movie"⟩. In general, people would only recommend something they like. Thus, we can do some reasoning and derive the label of an input "I would recommend this movie."

Under this formalism, it is trivial that perfect LLMs can follow instructions and solve the task because Assumption 2.2 ensures that a perfect LLM only generates logically consistent completion. The intricate question is, how is it possible for an LM to generalize from the training corpus to unseen instructions? We discuss this more in §4.

3 Bounding ICL Loss

We demonstrate how we can use our framework to analyze ICL. By adapting and combining the analyses by Zhang et al. (2023) and Hahn and Goyal (2023), we have the following theorem:

Theorem 3.1 (Average ICL Loss)**.** *Let the description of a classification task be $\{z_y\}_{y\in\mathcal{Y}}$ and z^* represent that the task descriptions $\{v_y\}_{y\in\mathcal{Y}}$ are associated with the corresponding verbalizers $\{r_y\}_{y\in\mathcal{Y}} \subset \Gamma$ used for ICL. Let K be the constraints used for decoding, and $\dot{g}$ be the event where a document follows certain formats. Let $S_t = x_1, r_2, d, x_2, r_2, \cdots, x_t, r_t, d$, where r_t is the verbalizer that is associated with the label of x_t and d is the delimiter. We have for any integer $T > 0$, the average cross-entropy loss of ICL is bounded as:*

$$\begin{aligned}
&-\frac{1}{T}\sum_{t=0}^{T}\log \Pr(r_t|x_t, S_{t-1}, K) \\
&\leq -\frac{1}{T}\log \Pr(z^*, \dot{g}|K) \\
&\quad -\frac{1}{T}\sum_{i=1}^{T}\log \Pr(r_t, d|x_t, z^*, \dot{g}, S_{t-1}, K) \\
&\quad -\frac{1}{T}\sum_{i=1}^{T}\log \frac{\Pr(x_t|z^*, \dot{g}, S_{t-1}, K)}{\Pr(x_t|S_{t-1}, K)}
\end{aligned} \tag{1}$$

When the last two terms on the right-hand side are non-negative, Eq. 1 shows the average cross-entropy loss of ICL converges to 0 in $\mathcal{O}(1/T)$. We discuss the terms on the right-hand side below.

The second term becomes 0 if we set K as the constraint that the next two tokens of S_{t-1}, x_t must be a verbalizer and the delimiter for all t. This is because Assumption 2.2 ensures that x_t does not conflict with r_t and in general, $\{v_y\}_{y\in\mathcal{Y}}$ conflict with each other, so r_t is the only valid continuation.

We then look at the last term. This term is 0 when x_t is conditionally independent to z^* as assumed by Zhang et al. (2023). However, this may be an over-simplification because, in natural language, the transition from x_t to its next token depends on the content of x_t. Fortunately, this assumption may actually be unnecessary for convergence because x_t is an example from a downstream task related to z^*; it is likely that

$$\Pr(x_t|z^*, \dot{g}, S_{t-1}, K) \geq \Pr(x_t|S_{t-1}, K),$$

which implies that this term is non-negative, and we can thus ignore this term. More rigorously, we can show the following corollary:

Corollary 3.2 (Expected Average ICL Loss)**.** *Let $\dot{g}$ represent a set of documents whose paragraphs are conditionally independent to each other given z^*, i.e., $\Pr(x_1, d, x_2, d, \cdots, x_T, d|z^*, \dot{g}) = \prod_{t=1}^{T}\Pr(x_t, d|z^*, \dot{g})$. If the downstream task data distribution $\mathcal{D}_X$ follow $\Pr(x|z^*, \dot{g})$, then we can bound the average ICL cross-entropy loss over the downstream task as:*

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x_1, x_2, \cdots, x_T \sim \mathcal{D}_X^T}\left[-\frac{1}{T}\sum_{t=0}^{T}\log \Pr(r_t|x_t, S_{t-1}, K)\right] \\
&\leq -\frac{1}{T}\log \Pr(z^*, \dot{g}|K).
\end{aligned} \tag{2}$$

The right-hand side of Eq. 2 characterizes the convergence rate and reflects the difficulty of doing ICL. If $z^*, \dot{g}, K$ are independent, then we can see this term is proportional to $\Pr(z^*)$. This implies that when the association between label description and the verbalizer is uncommon in the training data (e.g., associating "positive" to "This movie is bad."), doing ICL is more difficult.

Eq. 1 also allows us to analyze the scenario where we do not constrain the next token of

S_{t-1}, x_t to be a verbalizer while decoding. We can replace $\dot{g}$ with $\ddot{g}$ that represents the documents satisfying $\dot{g}$ and the constraint K (i.e., verbalizers always follow x).[2] This ensures the second term of Eq. 1 to be 0. The cost of using $\ddot{g}$ is that the first term in the bound gets greater because $\Pr(\ddot{g}|z^*) \leq \Pr(\dot{g}|z^*)$. This reflects that doing ICL without constraining the next token is harder.

$\Pr(\ddot{g}|z^*)$ may also be related to instruction tuning. The training examples for instruction tuning are input-output pairs following some format, such as having an instruction (e.g., "Label the example as positive if ...") at the beginning and a prompt after each input (e.g., "`[example]`. The sentiment of this comment is"). Because for examples that follow the special format, the next token after the prompt is always a verbalizer, these examples belong to the genera $\ddot{g}$. Having these examples in the training set would increase $\Pr(\ddot{g}|z^*)$ and thus make the ICL loss bound converge faster. This explains why instruction tuning helps ICL.

Note that we can extend the results to generation tasks. For generation tasks, we usually use a separator (or a short span of text) between the x_t and r_t. We can see the separator as an expression that can be associated with different senses, so the latent space for z is the senses the separator can be associated with, and z^* means that it is associated with the task instruction. In this way, we can apply our analysis to generation tasks.

4 Generalization

Assuming a latent model poses a dilemma: Language can encode various meanings, so assuming that the latent space is finite is unreasonable unless the space is very large. However, if the latent space is infinite or is very large, it is possible that the limited training data does not cover the whole space. Without any assumption on the latent space (e.g., the relation between the states in the space), it is impossible to discuss the generalization to unseen latent states. Thus, we provide an extension to our theoretical framework:

Assumption 4.1 (Meaning representation)**.** There exists (1) a finite set of *atom concepts* Ω, (2) a knowledge base KB consisting of logical rules between the atom concepts in Ω, and (3) a function f that can map any sentence in language to its meaning represented as a logical formula with operands in Ω such that for any two sentences s_1, s_2, the logical relation between s_1 and s_2 judged by humans is the same as $f(s_1)$ and $f(s_2)$ given the rules in the knowledge base KB.

The three items in this assumption corresponds to theories in various fields. The notion of atom concepts is aligned cognitive psychology studies that hypothesize the existence of a set of mental tokens (Fodor, 1975, 2008). and a recent study (Piantadosi, 2021) suggesting that semantics can be encoded with the combination of only a few symbols. The notion of knowledge base follows the early formulation of AI (Siskind, 1996; Murphy, 2004). As for the existance of a parsing function f, it follows the long history of linguistics studying the relationships between natural languages and formal languages (Carnap et al., 1968; Bresnan and Bresnan, 1982; Steedman, 1987, 1996; Sag et al., 1999), such as first-order logic (Frege et al., 1879; Peirce, 1883).

This assumption suggests that if we have the parsing function f, solving NLP tasks only requires a finite-length program that can do logical reasoning by manipulating logical symbols according to logical induction rules. If a deep model can learn this program, then it can perform a task even if this task is not in the training data. This assumption of a finite Ω also instantiates the concept of "language skills" by Arora and Goyal (2023), and their theoretical results are thus applicable.

5 Relating to Description Length

When there are no decoding constraints, we may see $\Pr(r_t, d|x_t, z^*, S_{t-1})$ as the difficulty of the example. To see this, we need an additional assumption:

Assumption 5.1. In some documents in the training data, the paragraphs are constituted with steps in a logical induction process, with some steps randomly dropped.

This kind of document may be pervasive in the training data. Essays arguing some claims are one example. To be convincing, these essays should proceed like a proving process that induces their conclusions. Documents describing a series of events can be another example, as events follow commonsense and develop progressively.

With this assumption and some regularity assumptions on the data distribution, we can have

$$\Pr(r_t, d|x_t, z^*, S_{t-1}) \leq c \cdot \ell(x_t), \qquad (3)$$

[2] Although $\ddot{g}$ may seem unnatural, this genre of documents corresponds to the PCFG structure assumed in Hahn and Goyal (2023).

Train	x57 x56 x64 r3 → x79 , r1 x57 → x58 , x90 x58 → r3 ... ; x80 x66 x63 x83 x1 → x82 , x80 x82 → r1 , ... , x64 x80 → x54 .
ICL	x44 x67 x34 x62 → **r2** ; x55 x38 x50 x48 → **r1** ; x21 x59 x57 x86 → **r2** ; x55 x76 x84 x99 →
CoT	x44 x67 x34 x62 → x16 , x34 x62 → x99 , x99 x16 → **r1** ; x77 x34 x62 x97 → x12 ... ; x21 x59 x57 x86 → x69 , x59 x57 → x75 , x69 x75 → **r2** ; x55 x76 x84 x99 →

Table 1: Calcutec examples for training, in-context learning (ICL), and chain-of-thought (CoT). The symbols in the bold font are the verbalizers in our synthetic setup.

where $\ell(x_t)$ is the number of reasoning steps required to solve the task, and c is a constant This $\ell(x_t)$ corresponds to the description length of the function that maps the inputs to their label in the loss bound by Hahn and Goyal (2023) (more discussion in Appendix E).

6 Empirical Experiments

We present two synthetic setups to demonstrate that LMs can acquire ICL capability (§6.1) and instruction following capability (§6.2) from a training dataset built according to our framework. Finally, in §6.3, we present real world evidence that supports our theory.

6.1 Inspecting the ICL Capability

We present a synthetic setup, Calcutec, as a concrete instantiation of our theoretical framework. With Calcutec, we show that Transformers can acquire ICL capability by modeling the linguistic characteristics specified in our framework.

6.1.1 Calcutec

Setup Following our framework in §2 and §4, we construct a pseudo-language:

- Logic model: We use a subset of propositional logic as our logic model. We only consider Horn clauses (Horn, 1951), i.e., formulas in the form $A \wedge B \rightarrow C$.
- Atom concepts: We use 100 symbols as our set of atom concepts Σ.
- KB: We generate a knowledge base by generating 5 formulas of the form $\sigma_1 \wedge \sigma_2 \rightarrow \sigma$ for each $\sigma \in \Sigma$, where σ_1, σ_2 are sampled from $\Sigma \backslash \{\sigma\}$ uniformly at random.
- We have a set $\Gamma = \{r_i\}_{i=1}^4$ representing the expressions described in Assumption 2.3.

Training Dataset. Following Assumption 2.4, a document is a concatenation of paragraphs separated by delimiter ";" and ends with ".". In our synthetic language model training dataset, each document contains 16 paragraphs.

Because sentences in the real world are not ordered arbitrarily, we follow Assumption 5.1 and generate random paragraphs following the structure of logical proofs. Each paragraph represents the induction process of $P \models g$ for some randomly selected $P \subset \Sigma$ and $g \in \Sigma$. Each sentence in the paragraph is a sentence representing a reasoning step. We separate the clauses in the sequence with commas. To simulate the fact that documents in the real world always skip some reasoning steps, we further apply some perturbations over the generated paragraphs that drop some reasoning steps with a skip rate p_{skip}. After we generate a document, we replace some symbols in the document with expression $r_a, r_b \in \Gamma$ (details in Appendix F and the pseudo-code Alg. 1). [3]

Downstream Tasks. Following the formalism in §2.2, we define a binary classification task by defining the descriptions v_+ and v_- of the positive and negative classes, respectively. We use the disjunctions of atom concepts (i.e., in the form of $a_1 \vee a_2 \vee \cdots$) as the descriptions of classes. We create five downstream tasks using different disjunctions. Each input is a subset of variables in Σ from which we ensure that only one of the classes can be induced.

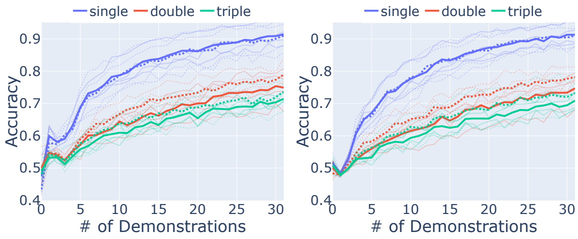
Demonstration. We represent an input-label pair as $x_1 x_2 \cdots \rightarrow r$, where $x_1 x_2 \cdots$ is the input part and $r \in \{r_+, r_-\} \subset \Gamma$ is an expression in Γ serving as the verbalizer.

Chain-of-thought. A chain-of-thought is in the format same format as the training data, but ends with an expression $r \in \{r_+, r_-\}$, e.g., $x_1 x_2 \cdots \rightarrow$

[3] Models can acquire in-context learning ability even with $p_{skip} = 0$ (Figure 6 in thr appendix).

Task	r_1, r_2 ICL	r_1, r_2 CoT	r_3, r_4 ICL	r_3, r_4 CoT
Single	57.1	91.7	55.6	92.0
Double	53.5	76.3	51.1	77.1
Triple	53.0	73.0	51.7	73.4

Table 2: The 4-shot accuracy of ICL versus chain-of-thought (CoT) using different verbalizers.



(a) Verbalizers = r_1, r_2 (b) Verbalizers = r_3, r_4

Figure 1: In-context learning accuracy with Calcutec when using different verbalizers (r_1, r_2 or r_3, r_4). The dotted lines represent the performance of *unseen combinations* described in §6.1.2. The colors represent the number of atom concepts each class (v_+ or v_-) is associated with. The main lines represent the average accuracy of 5 tasks. Lines in the lighter color represent the individual tasks.

$x_3; x_3 \cdots x_4 \rightarrow r_+$. This chain-of-thought reflects the step-by-step induction process from the inputs to the label.

6.1.2 Distribution Shifts

We make experimental designs to simulate the real-world distribution shifts from training to inference:

Format Mismatch. The reasoning steps are present in the training data but not in the prompts.

Verbalizer Mismatch. When we are picking the expressions in Γ, we assign probability mass 45%, 45%, 5%, 5% to r_1, r_2, r_3, r_4. In this way, we can inspect the effect of using less frequent verbalizers.

Unseen Tasks. To investigate whether the model can generalize to a new combination of formulas unseen in the training data when we generate our training data, we ensure that the expressions in Γ are only associated with the disjunctions of two atom concepts s_1, s_2 from a strict subset of all the possible combinations $\Sigma \times \Sigma$. We then test the trained model on tasks where v_+ and v_- are the disjunctions of the unseen combinations. We also test the models on tasks where v_+ and v_- are the disjunctions of three atom concepts $\in \Sigma \times \Sigma \times \Sigma$.

6.1.3 Experiment Details

We use $p_{skip} = 0.25$ in our experiment. We generate 60,000 documents with 16 paragraphs, as described above. Among them, we use 10k for validation. We train a 6-layer Transformer (Vaswani et al., 2017) model until the loss on the validation set converges. We include additional setups in §H.

6.1.4 Results and Discussion

Figure 1 shows that the model trained with Calcutec can perform in-context learning This evidence supports our Pelican Soup framework. We further inspect the ICL performance under the distribution shifts described in §6.1.2:

- Infrequent verbalizer: We observe similar performance regardless of the frequency of the verbalizers (r_1, r_2 versus r_3, r_4).
- Unseen tasks: Figure 1 shows that the model has similar performance over tasks defined with unseen and unseen combinations of atom concepts (dot lines and solid lines). The models can also generalize to tasks defined with three latent concepts (green lines).

In sum, the results show that the model can generalize well under several distributional shifts.

We also experiment with 4-shot learning using chain-of-thought. Table 2 shows that the model also benefits from chain-of-thought. We conjecture that it is because chain-of-thought has a format more similar to the format for training.

6.2 Digit Addition Task

In addition to the ICL capability we have inspected in §6.1, we will also inspect the instruction-following capability of LMs. To this end, we present a digit addition task. The goal is to inspect whether models can generalize to unseen instructions by modeling the knowledge of the interrelation between senses exhibited in the step-by-step reasoning process.

6.2.1 Setup

We utilize the algebraic structure of the integers under addition to construct a language where each expression is constantly associated with a sense. In this language, a paragraph is the digit-by-digit process of solving an addition task based on mathematical rules. The rules exhibited in each of the

Training	4 9 2 8 4 6 + 0 8 0 3 5 0 = 0 0 0 0 0 0 ; 0 9 2 8 4 6 + 0 8 0 3 5 0 = 4 0 0 0 0 0 ; 0 0 2 8 4 6 + 0 0 0 3 5 0 = 4 7 1 0 0 0 ; · · · 0 0 0 0 0 0 + 0 0 0 0 0 0 = 4 7 3 1 0 7 ;
Testing	8 7 4 0 1 6 + 0 9 2 1 5 0 = 0 0 0 0 0 0 ; 0 0 0 0 0 0 + 0 0 0 0 0 0 =

Table 3: Training and testing examples for the digit addition task.

steps reflect the interrelation between the sense of the expressions.

We generate a training set consisting of digit-by-digit reasoning processes for 100,000 pairs of numbers. In our training set, we drop each intermediate step at a constant probability of p_{drop}. After training a Transformer model with the language modeling objective, we test whether the model can generate the final answer without generating the intermediate steps for unseen number pairs.

We show examples of our data in Table 3. Each digit sequence represents a number from the lower digit to the higher digit. The reasoning process in the training set gradually updates both sides of "=" from the lowest digit to the highest digit. As for the testing example, we skip the intermediate steps, prompting the model to complete the right-hand side of "=" in the last step. We include a rigorous description in Appendix J.

6.2.2 Results and Discussion

We report the exact match accuracy and the digit-level accuracy of models trained with different p_{drop} in Figure 2 with 5 random seeds. A higher accuracy implies the model generalizes better from step-by-step reasoning processes. The results show that three of the four models can achieve perfect accuracy when p_{drop} is as small as 0.3 but achieve an accuracy less than 0.2 when $p_{drop} = 0.9$. It suggests that models can follow prompts by modeling the inter-expression relation exhibited in the step-by-step reasoning process.

Additionally, we observe that larger models tend to have higher and more stable accuracy. When the number of digits is 6 (Figure 14), only the largest model can achieve perfect accuracy. This observation is aligned with the emergence of large language models' ability.

6.3 Real-world Evidence

We inspect whether LMs can do ICL with pronouns well because pronouns are reference words frequently associated with different meaning and our framework suggests that LMs learn ICL ability by modeling the association between reference words and their meaning. We thus experiment with the template "`[input]`", `[verbalizer]` *thought.* and use "he", "she" as the verbalizers. We follow the setup in Min et al. (2022a) and compare the accuracy of the binary classification tasks, including SST-2 (Socher et al., 2013), CR (Hu and Liu, 2004), MR (Pang and Lee, 2005), and Subj (Pang and Lee, 2004), using GPT2-Large.

task	direct	pronoun
SST-2	63.0	65.3
CR	61.7	62.9
MR	59.2	56.7
Subj	51.0	62.2

Table 4: The accuracy of using task-specific templates/verbalizers (direct) (Min et al., 2022a) v.s. using task-agnostic templates/pronouns for 16-shot in-context learning with GPT2-Large.

Table 4 shows that this task-agnostic template with pronouns is competitive with those task-specific templates. This contradicts the belief that only larger models can do in-context learning with task-irrelevant verbalizer Wei et al. (2023). It suggests that modeling reference-meaning may indeed contribute to LMs' ICL ability.

7 Related Work

Since Brown et al. (2020) discovered large language models' in-context learning ability, some theoretical works have attempted to explain how language models acquire this ability. Based on a hidden Markov model (HMM) assumption on the language generation process, Xie et al. (2022) suggested that in-context learning is an implicit Bayesian inference process. Hahn and Goyal (2023) defined the generation process with Compositional Attribute Grammar, which is weaker than the HMM assumption, explaining the in-context learning ability with the minimum description length. They also studied the compositionality of natural language tasks with function compositions. Zhang et al. (2023) assumed a more general latent variable model. Arora and Goyal (2023) ana-

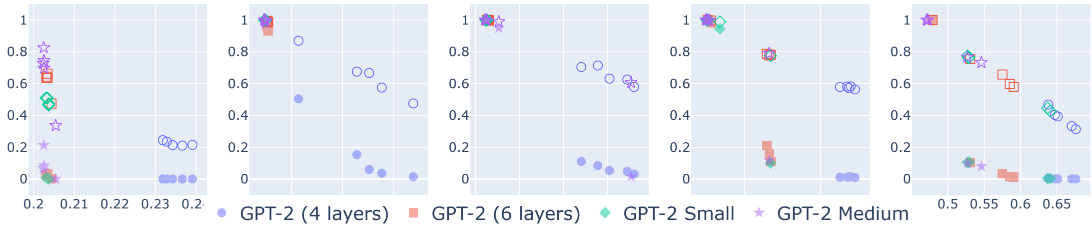


Figure 2: The exact accuracy (y-axis, solid points) and digit-level accuracy (y-axis, hollow points) versus validation loss (x-axis) for the 5-digit addition task for dropping rates $p_{drop} = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ (from left to right) for five random seeds (points in each figure). We provide more results in Figure 13 and Figure 14 in the appendix.

lyze the emergence of skills based on the scaling law (Hoffmann et al., 2022). While their analysis assumes a set of atomic skills for NLP tasks, our framework is based on a set of atom concepts.

There were also many empirical studies on the in-context learning ability. Some works focused on the effect of the instruction (Webson and Pavlick, 2022; Lampinen et al., 2022; Jang et al., 2023), while some focused on the examples in the demonstration (Liu et al., 2022; Lu et al., 2022; Sorensen et al., 2022; Min et al., 2022b; Yoo et al., 2022; Ye et al., 2023; Chang and Jia, 2023; Ye et al., 2023; Wang et al., 2023b; Kossen et al., 2023). Shin et al. (2022) found that not all training corpora led to in-context learning ability. Prystawski and Goodman (2023) used synthetic data to suggest that the pretraining dataset's locality structure contributes to the reasoning steps' effectiveness. Wang et al. (2023a) studied the reasoning steps in chain-of-thought. Akyürek et al. (2024) formulated ICL as learning a formal language from demonstrations and benchmarked model families.

Some previous work studied in-context learning as a meta-learning-like problem (Chen et al., 2022). Some works focused on the relationships between in-context learning and optimization algorithms (Garg et al., 2022; von Oswald et al., 2022; Akyürek et al., 2023; Fu et al., 2023; Guo et al., 2023). Some works inspected the mechanism of ICL in transformer models (Hendel et al., 2023; Bietti et al., 2023; Todd et al., 2023; Shen et al., 2023; Bai et al., 2023). Chan et al. (2022) studied the properties of dataset distribution that could contribute to the in-context learning ability. Li et al. (2023b) provided generalization bounds based on the stability of Transformer models and the distance of downstream tasks. We instead focus on how the pretraining data in natural language contributes to the ICL learning ability.

8 Conclusion and Future Work

In this work, we propose a framework that explains how linguistic phenomena in the training corpus lead to LLMs' ICL and instruction-following capability. Compared with previous works (Xie et al., 2022; Zhang et al., 2023), our latent model better reflects the complexity of language. By introducing the notion of knowledge base and logic system, our framework provides insights into how LLMs can generalize from pretraining to downstream tasks, instantiating a setup compatible with the assumptions made by Arora and Goyal (2023). We also relate our bound to the function description length discussed by Hahn and Goyal (2023).

Our framework illuminates a few possible directions for improving LLMs:

1. Our work highlights the importance of learning the interrelation between senses. As previous works have shown that the language modeling objective is inefficient for this purpose (Allen-Zhu and Li, 2023; Chiang et al., 2024), we suggest that developing a more sophisticated learning algorithm is crucial.

2. Our theory suggests that LLMs' generalization depends on the models' ability to parse sentences into logical representations. Thus, evaluating and improving LLMs' semantic parsing ability may be a promising direction.

3. The experimental results of our addition tasks indicate a curious ability of Transformer models: Transformer models can generalize to unseen prompts by modeling the intermediate step-by-step reasoning process. This may be related to the success of the symbolic chain-of-thought distillation (Li et al., 2023a; Hsieh et al., 2023; Shridhar et al., 2023). Investigating the mechanism and reinforcing this ability may improve the efficiency of LM training.

9 Limitations

A limitation of our framework is that, as most theoretical studies do, we simplify the real-world scenario to draw insights. One simplification we make is that, we do not take the noise in LLMs' training data into account. While our preliminary experiment with the digit addition task in §K show that LMs can acquire the zero-shot instruction following capability even when the training data is noisy, we still need to make more assumption on the noise to establish a generic theoretical result. We thus leave it for future study. Another simplification is that, we assume that the language model can perfectly model the distribution of natural language. However, it is unlikely to be the case in practice. On the one hand, the training data may not cover all the test cases. On the other hand, LLMs may not generalize perfectly from the training set. We need to make more assumption on the training/test data distribution and/or having deeper understanding on how deep learning models generalize to alleviate this assumption. Therefore, we deem it out of the scope of this paper.

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*.

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. 2024. In-context language learning: Arhitectures and algorithms. *arXiv preprint arXiv:2401.12973*.

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*.

Sanjeev Arora and Anirudh Goyal. 2023. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. 2023. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*.

Alberto Bietti, Vivien Cabannes, Diane Bouchacourt, Herve Jegou, and Leon Bottou. 2023. Birth of a transformer: A memory viewpoint. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Joan Bresnan and Joan Wanda Bresnan. 1982. *The mental representation of grammatical relations*, volume 1. MIT 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.

Rudolf Carnap et al. 1968. *Logische syntax der sprache*. Springer.

Stephanie C. Y. Chan, Adam Santoro, Andrew K. Lampinen, Jane X. Wang, Aaditya Singh, Pierre H. Richemond, Jay McClelland, and Felix Hill. 2022. Data distributional properties drive emergent in-context learning in transformers.

Ting-Yun Chang and Robin Jia. 2023. Data curation alone can stabilize in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8123–8144, Toronto, Canada. Association for Computational Linguistics.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730, Dublin, Ireland. Association for Computational Linguistics.

Ting-Rui Chiang, Xinyan Yu, Joshua Robinson, Ollie Liu, Isabelle Lee, and Dani Yogatama. 2024. On retrieval augmentation and the limitations of language model training. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 229–238, Mexico City, Mexico. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Jerry A. Fodor. 1975. *The Language of Thought*. Harvard University Press.

Jerry A. Fodor. 2008. *LOT 2: The Language of Thought Revisited*. Oxford University Press.

Gottlob Frege. 1879. Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought [1879]. *From Frege to Gödel: A Source Book in Mathematical Logic*, 1931:1–82.

Gottlob Frege. 1948. Über sinn und bedeutung. *Philosophical Review*, 57(a).

Gottlob Frege et al. 1879. Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought. *From Frege to Gödel: A source book in mathematical logic*, 1931:1–82.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. 2023. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. *arXiv preprint arXiv:2310.17086*.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*.

Tianyu Guo, Wei Hu, Song Mei, Huan Wang, Caiming Xiong, Silvio Savarese, and Yu Bai. 2023. How do transformers learn in-context beyond simple functions? a case study on learning with representations. *arXiv preprint arXiv:2310.10616*.

Michael Hahn and Navin Goyal. 2023. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

Alfred Horn. 1951. On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, 16(1):14–21.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, Toronto, Canada. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA. Association for Computing Machinery.

Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2023. Can large language models truly understand prompts? a case study with negated prompts. In *Proceedings of The 1st Transfer Learning for Natural Language Processing Workshop*, volume 203 of *Proceedings of Machine Learning Research*, pages 52–62. PMLR.

Jannik Kossen, Tom Rainforth, and Yarin Gal. 2023. In-context learning in large language models learns label relationships but is not conventional learning. *arXiv preprint arXiv:2307.12375*.

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

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023a. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2665–2679, Toronto, Canada. Association for Computational Linguistics.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. 2023b. Transformers as algorithms: Generalization and stability in in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19565–19594. PMLR.

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

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

John McCarthy. 1960. Programs with common sense.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. 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.

Gregory Murphy. 2004. *The big book of concepts*. MIT press.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Charles S Peirce. 1883. *A theory of probable inference*. Little, Brown and Co.

Steven T. Piantadosi. 2021. The computational origin of representation. *Minds Mach.*, 31(1):1–58.

Ben Prystawski and Noah D Goodman. 2023. Why think step-by-step? reasoning emerges from the locality of experience. *arXiv preprint arXiv:2304.03843*.

Paul Rottger, Bertie Vidgen, Dirk Hovy, and Janet Pierrehumbert. 2022. Two contrasting data annotation paradigms for subjective NLP tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 175–190, Seattle, United States. Association for Computational Linguistics.

Ivan A Sag, Thomas Wasow, Emily M Bender, and Ivan A Sag. 1999. *Syntactic theory: A formal introduction*, volume 92. Center for the Study of Language and Information Stanford, CA.

Roger C Schank and Robert P Abelson. 1988. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.

Lingfeng Shen, Aayush Mishra, and Daniel Khashabi. 2023. Do pretrained transformers really learn in-context by gradient descent? *arXiv preprint arXiv:2310.08540*.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, and Nako Sung. 2022. On the effect of pretraining corpora on in-context learning by a large-scale language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5168–5186, Seattle, United States. Association for Computational Linguistics.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073, Toronto, Canada. Association for Computational Linguistics.

Jeffrey Mark Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1):39–91. Compositional Language Acquisition.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Taylor Sorensen, Joshua Robinson, Christopher Rytting, Alexander Shaw, Kyle Rogers, Alexia Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. 2022. An information-theoretic approach to prompt engineering without ground truth labels. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 819–862, Dublin, Ireland. Association for Computational Linguistics.

Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5(3):403–439.

Mark Steedman. 1996. *Surface structure and interpretation*, volume 30. MIT press Cambridge, MA.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2023. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*.

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.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2022. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Xinyi Wang, Wanrong Zhu, and William Yang Wang. 2023b. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*.

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.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Veselin Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2023. Complementary explanations for effective in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4469–4484, Toronto, Canada. Association for Computational Linguistics.

Kang Min Yoo, Junyeob Kim, Hyuhng Joon Kim, Hyunsoo Cho, Hwiyeol Jo, Sang-Woo Lee, Sang-goo Lee, and Taeuk Kim. 2022. Ground-truth labels matter: A deeper look into input-label demonstrations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2422–2437, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. 2023. What and how does in-context learning learn? bayesian model averaging, parameterization, and generalization. *arXiv preprint arXiv:2305.19420*.

A Motivation of the Pelican Soup Framework

The Pelican Soup game inspires our framework. It is a game involving a puzzle master who has a story in their mind. The game participants aim to recover the story by asking the puzzle master yes/no questions. An observation is that once the participants recover the story, they can answer any questions about the story. Therefore, the story has a similar role as a latent variable defining the input-output mapping, and the yes/no questions are similar to the demonstrations for in-context learning. We include an example in Appendix B.

Given the above observation, we can study in-context learning by considering why humans can solve Pelican Soup riddles. We conjecture that this is because the person who makes the story and the ones who solve the riddle share the same (or similar) commonsense (McCarthy, 1960) about logical relationships among things in this world (Schank and Abelson, 1988). This inspires us to introduce the notion of a commonsense knowledge base in our framework.

B A Pelican Soup Example

Puzzle master: A men walks into a restaurant and orders pelican soup. After taking a sip, he loses his mind. Why?

Participants: Is it because the soup is not cooked well?

Puzzle master: No.

Participants: Is it because the soup toxic?

Puzzle master: No.

Participants: Does the soup remind him something?

Puzzle master: Yes.

Participants: Did someone cook pelican soup for him?

Puzzle master: Yes.

Participants: Is that person still alive?

Puzzle master: No.

For the sake of aesthetics, we do not include the latent story here. If you are interested, please check it online.

C Proof of Theorem 3.1

Let $S_t = x_1, r_2, d, x_2, r_2, d \cdots, x_t, r_t, d$.

$$
\begin{aligned}
&\Pr(z, g|S_t, K) \\
=&\frac{\Pr(S_t|z, g, K)\Pr(z|K)}{\sum_z \Pr(S_t|z, g, K)\Pr(z, g|K)} \\
=&\frac{\Pr(z, g|K)\prod_{i=1}^{t}\Pr(x_i, r_i, d|z, g, S_{i-1}, K)}{\sum_{z', g}\Pr(z', g|K)\prod_{i=1}^{t}\Pr(x_i, r_i, d|z', g, S_{i-1}, K)}
\end{aligned}
$$

$$
\begin{aligned}
&P(x_{t+1}, r_{t+1}, d|S_t, K) \\
=&\sum_{z,g}\Pr(x_{t+1}, r_{t+1}, d|z, S_t, K)\Pr(z, g|S_t, K) \\
=&\frac{\sum_{z,g}\Pr(z, g|K)\prod_{i=1}^{t+1}\Pr(x_i, r_i, d|z, g, S_{i-1}, K)}{\sum_{z'}\Pr(z, g|K)\prod_{i=1}^{t}\Pr(x_i, r_i, d|z', g, S_{i-1}, K)}.
\end{aligned}
$$

Thus, it holds that

$$
\begin{aligned}
&-\sum_{t=0}^{T}\log \Pr(x_{t+1}, r_{t+1}, d|S_t, K) \\
=&-\sum_{t=0}^{T}\Big(\\
&\log\sum_{z,g}\Pr(z,g|K)\prod_{i=1}^{t+1}\Pr(x_i, r_i, d|z, g, S_{i-1}, K) \\
&-\log\sum_{z,g}\Pr(z,g|K)\prod_{i=1}^{t}\Pr(x_i, r_i, d|z, g, S_{i-1}, K) \\
&\Big) \\
=&-\log\sum_{z,g}\Pr(z,g|K)\prod_{i=1}^{T+1}\Pr(x_i, r_t, d|z, g, S_{i-1}, K) \\
\leq&-\log\Pr(z^*,\dot{g}|K)\prod_{i=1}^{T+1}\Pr(x_i, r_t, d|z^*, \dot{g}, S_{i-1}, K) \\
=&-\log\Pr(z^*,\dot{g}|K) \\
&-\sum_{i=1}^{T+1}\log\Pr(x_i, r_i, d|z^*, \dot{g}, S_{i-1}, K) \\
=&-\log\Pr(z^*,\dot{g}|K) \\
&-\sum_{i=1}^{T}\log\Pr(r_i, d|x_i, z^*, \dot{g}, S_{i-1}, K) \\
&-\sum_{i=1}^{T}\log\Pr(x_i|z^*, \dot{g}, S_{i-1}, K).
\end{aligned}
$$

Thus,

$$
\begin{aligned}
&-\frac{1}{T}\sum_{t=0}^{T}\log\Pr(r_t|x_t, S_{t-1}, K) \\
\leq&-\frac{1}{T}\Big(\log\Pr(z^*,\dot{g}|K) \\
&+\sum_{i=1}^{T}\log\Pr(r_t, d|x_t, z^*, \dot{g}, S_{t-1}, K) \\
&+\sum_{i=1}^{T}\log\frac{\Pr(x_t|z^*, \dot{g}, S_{t-1}, K)}{\Pr(x_t|S_{t-1}, K)}) \\
&+\frac{1}{T}\sum_{i=1}^{T}\log\Pr(d|r_t, x_t, S_{t-1}, K\Big). \\
\leq&-\frac{1}{T}\Big(\log\Pr(z^*,\dot{g}|K) \\
&+\sum_{i=1}^{T}\log\Pr(r_t, d|x_t, z^*, \dot{g}, S_{t-1}, K) \\
&+\sum_{i=1}^{T}\log\frac{\Pr(x_t|z^*, \dot{g}, S_{t-1}, K)}{\Pr(x_t|S_{t-1}, K)}\Big)
\end{aligned}
$$

D Proof of Corollary 3.2

The second term in the right-hand side of Eq. 1 is zero when the decoding constrain K is imposed. Therefore, it suffices to prove the last term is non-negative in expectation.

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{x_1,x_2,\cdots,x_T\sim\mathcal{D}_X^T}\sum_{i=1}^{T}\log\frac{\Pr(x_t|z^*,\dot{g},S_{t-1},K)}{\Pr(x_t|S_{t-1},K)} \\
=&\mathop{\mathbb{E}}_{x_1,x_2,\cdots,x_T\sim\mathcal{D}_X^T}\sum_{i=1}^{T}\log\frac{\Pr(x_t|z^*,\dot{g},K)}{\Pr(x_t|S_{t-1},K)} \\
=&\sum_{x_1,x_2,\cdots,x_T}\Pr(x_t|z^*,\dot{g},K)\sum_{i=1}^{T}\log\frac{\Pr(x_t|z^*,\dot{g},K)}{\Pr(x_t|S_{t-1},K)} \\
=&\mathrm{KLD}(\Pr(x_t|z^*,\dot{g},K)||\Pr(x_t|S_{t-1},K)) \geq 0
\end{aligned}
$$

E The Connection between $P(r_t|x_t, z^*, \ddot{g})$ and Function Description Length by Hahn and Goyal (2023)

Firstly, we make some regularity assumptions: Given a step-by-step reasoning process $\pi = s_1, s_2, \cdots, s_n$ for the induction process of $P \models Q$, in the training data,

1. each step may be dropped independently to each other with probability p_{drop}.

2. $\Pr(s_i|P, s_1, s_2, \cdots, s_{i-1}) > p_{min}$ for all $i \in [n]$.

We first show how we derive Eq. 3: Based on Assumption 5.1,

$$\Pr(r_t|x_t, z^*, \ddot{g}) = \sum_{\pi \in \Pi} \Pr(\pi, r_t|x_t, z^*, \ddot{g}) \Pr(\pi \text{ is dropped}),$$

where Π is a set of token sequences representing reasoning steps that induce r_t from x_t. Let π^* be the shortest proof in Π, we have

$$\begin{aligned} &\log \Pr(r_t|x_t, z^*, \ddot{g}) \\ =&\log \sum_{\pi \in \Pi} \Pr(\pi, r_t|x_t, z^*, \ddot{g}) \Pr(\pi \text{ is dropped}) \\ \geq& \log \Pr(\pi^*, r_t|x_t, z^*, \ddot{g}) \Pr(\pi^* \text{ is dropped}) \\ \geq& p_{min} \log \ell(\pi^*) + p_{drop} \log \ell(\pi^*). \end{aligned}$$

Then we can discuss the connection between $\Pr(r_t|x_t, z^*, \ddot{g})$ and the function description length by Hahn and Goyal (2023). We can see the dropped reasoning steps in π^* as the hidden (tree) structure that maps x_t to r_t as the derivation tree τ_ϕ in the bound of Hahn and Goyal (2023). The length of the reasoning steps thus corresponds to the description length of the derivation tree $\mathrm{D}(\tau_\phi)$.

A major difference between the bound by Hahn and Goyal (2023) and our bound is that their bound has $\mathrm{D}(\tau_\phi)$ constant to T while our bound has $\sum_t \log \Pr(r_t|x_t, z^*, \ddot{g})$, which potentially grows proportionally to T. The cause of this difference is that, Hahn and Goyal (2023) assumes a structure that repetitively applies a function mapping in a document, and the number of repetition is independent to the complexity of the function mapping. In comparison, our framework does not make this assumption.

F Detailed Gengeration Process of the LM Training Data in Calcutec

We generate a paragraph based on Assumption 2.3 in the following step:

1. We pick a symbol s from the symbols associated with r_a uniformly at random.

2. We randomly generate a proof for KB, $P \models g$, where $P \subset \Sigma$ is the premise and $g \in \Sigma$ is the goal of the proof. We ensure that this proof contains the topic s.

3. We convert the proof tree to a sequence of proving steps by traversing the proving tree in a topological order with ties broken randomly. Each node in the proof tree corresponds to a rule in KB, so the resulting sequence of proving steps consists of horn clauses in the form $a_1 a_2 \to b$. We separate the clauses in the sequence with commas.

4. We rewrite the first step of the proving process to contain the premises of the proof. Specifically, we replace the antecedent in the first formula with the premise P. We find that this step is necessary to prevent the language model trained on it from hallucinating irrelevant variables randomly. It is important for our experiment for chain-of-thought, but is not necessary for language models to learn the in-context learning ability.

G Perturbations in Calcutec

We apply two types of perturbations over the reasoning steps in Calcutec described in §6.1:

1. Random merge: At probability p_{merge}, for every two consecutive clauses where the consequence of the first one is in the antecedents of the second one, say $a_1 a_2 \to b_1$ and $b_1 a_3 \to b_2$, we merge them into a single clause $a_1 a_2 a_3 \to b_2$.

2. Random drop: Given a clause $a_1 a_2 \cdots a_n \to b$. We drop each of the antecedents $a \in \{a_1, a_2, \cdots a_n\}$ at probability p_{drop}. We apply this dropping to every clause in the proof except the first one to ensure that we do not drop the premises.

We use $p_{merge} = p_{drop} = p_{skip}$.

Additionally, when flatting the proof trees with topological sort, we break ties randomly. We also randomize the order of the symbols in the antecedents.

H Extra Experiments with Calcutec

H.1 Additional Setups

Unseen Inference Process. Based on Assumption 5.1 and the formalism of NLP tasks in §2.2, input-label pairs of a downstream task corresponds to prefix-reference pairs in a paragraph. To examine whether the trained model can generalize well when the induction process for the label is different

Algorithm 1 Pseudo code for the generation process of an Calcutec document used for training.

Sample r_a, r_b from $\{r_1, r_2, r_3, r_4\}$ with probability 0.45, 0.45, 0.05, 0.05.
Sample topic $S = \{s_1, s_2\} \subset \Sigma$.
Initialize a document D with empty string.
for $p = 1, 2, \ldots, n_{par}$ **do**
  while True **do**
    Sample $s \in S$.
    Sample a set $X \subset \Sigma$ such that $\bigwedge_{x \in X} x \models s$.
    Run the resolution algorithm to get the set $M = \{m | X \models m\}$.
    Find an extra premise x' that can increase the depth of deepest proof tree for $X \models m$.
    Run the resolution algorithm to get the set $M' = \{m | X \cup \{x'\} \models m\}$.
    if $|M'| > \frac{|\Sigma|}{2}$ **then**
      Reject the sampled $X \cup \{x'\}$. ▷ We don't want a premise that entails everything.
      Restart the while loop.
    end if
    Sample a $g \in M'$ such that the proof tree for $X' \models g$ contains s and its depth $> d_{min}$. ▷ We use $d_{min} = 4$ in our experiments.
    Do topological sort to flatten the proof tree and convert it into a string.
    Append the string to D.
  end while
end for
for $s \in S$ **do**
  $D \leftarrow D$.replace(s, r_a)
end for
Let $S' = \{s'_1, s'_2\} \in \Sigma$ be the top-2 frequent non r_a symbols in D.
for $s' \in S'$ **do**
  $D \leftarrow D$.replace(s', r_b)
end for

from the induction process for the pronoun in the training data, we generate a training dataset where all the pronouns are induced from the premise with a left-branching proof tree with a depth equal to 2 (Figure 4a), while the test data contains samples whose labels are induced from the input with balanced trees (Figure 4b).

Different Input Lengths. For each downstream tasks, we experiment with examples with different lengths. When the inference process is branching, having input length equal to 4 makes the proving tree deeper.

No perturbations. As described in §G, we apply some random perturbations on the proving process. We also experiment with the setup where we do not apply any perturbations.

With/Without Rewriting the First Step. As described in §F, we rewrite the first step of the proof. We also experiment with the setup where we do not rewrite the first step.

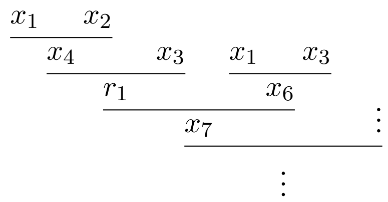
Model Size. We also experiment with different models sizes. We experiment with GPT-2 models that have 3, 4 and 5 layers.

H.2 Results and Discussion

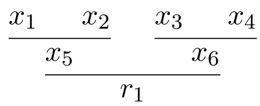
Unseen Inference Process. Figure 5a and Figure 5d show that the ICL performance on the branching examples is similar to the performance on the branching examples. It suggests that the model can generalize to examples that requires an unseen reasoning process. Interestingly, Table 5 show that using chain-of-thoughts mitigates this gap.

Different Input Lengths. Figure 5b and Figure 5e show that the model can still do ICL for the examples with length equal to 4. However, compared with the performance on examples with length equal to 3 (Figure 5c and Figure 5f), the performance is worse. This may be because solving these length-4 examples requires more reasoning steps.

With/Without Rewriting the First Step. Figure 8 shows that models trained with proofs that are rewritten has similar performance as models trained with the proofs that were rewritten (Figure 5). This suggests that rewriting the first step in the proof is not necessary for the model to acquire the ICL ability.



(a) The proof tree a paragraph in the training dataset corresponds .



(b) A balanced tree for a downstream task sample.

Figure 4: Proof trees examples.

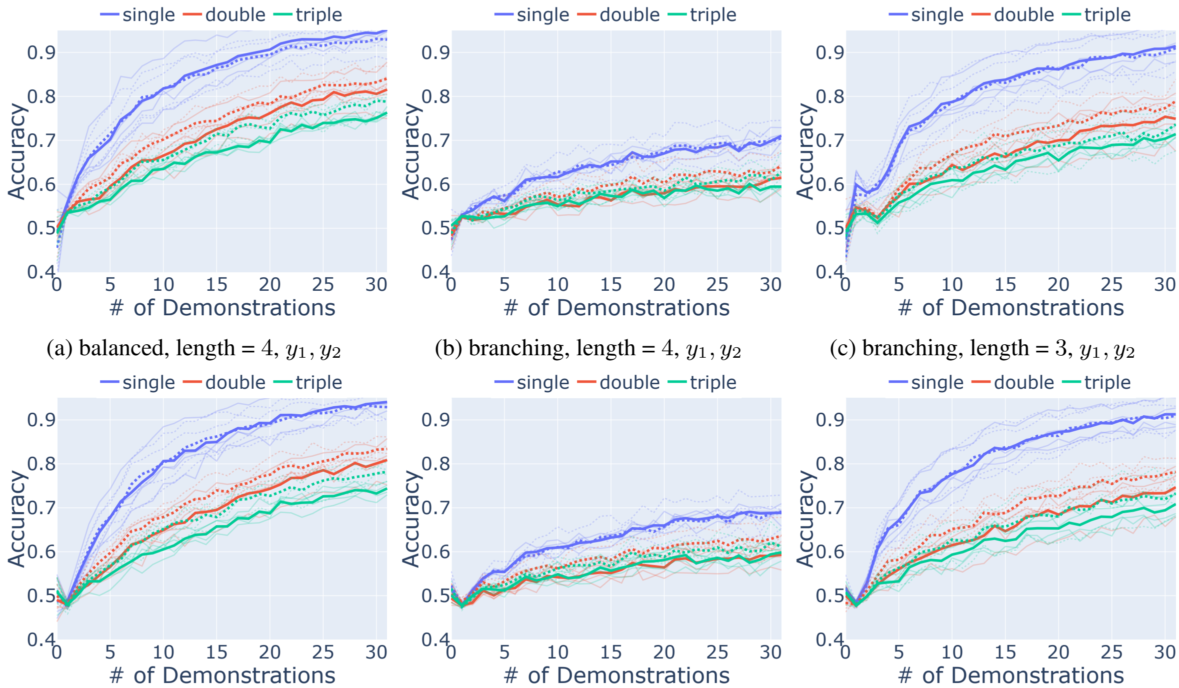
Model Size. Figure 9 show that deeper models have better ICL performance. It is aligned with the real-world observation that scaling helps model performance.

I Hyper-parameters

We train our model using batch size 256, warm up ratio 5%, and we truncate the sequence length to 512 tokens and the default parameters for the optimizer. We use the implementation of GPT-2 by Hugging Face transformers v4.27.2. All models can be trained with 4 RTX 2080ti within 8 hours.

J Formal Description of the Digit Addition Data

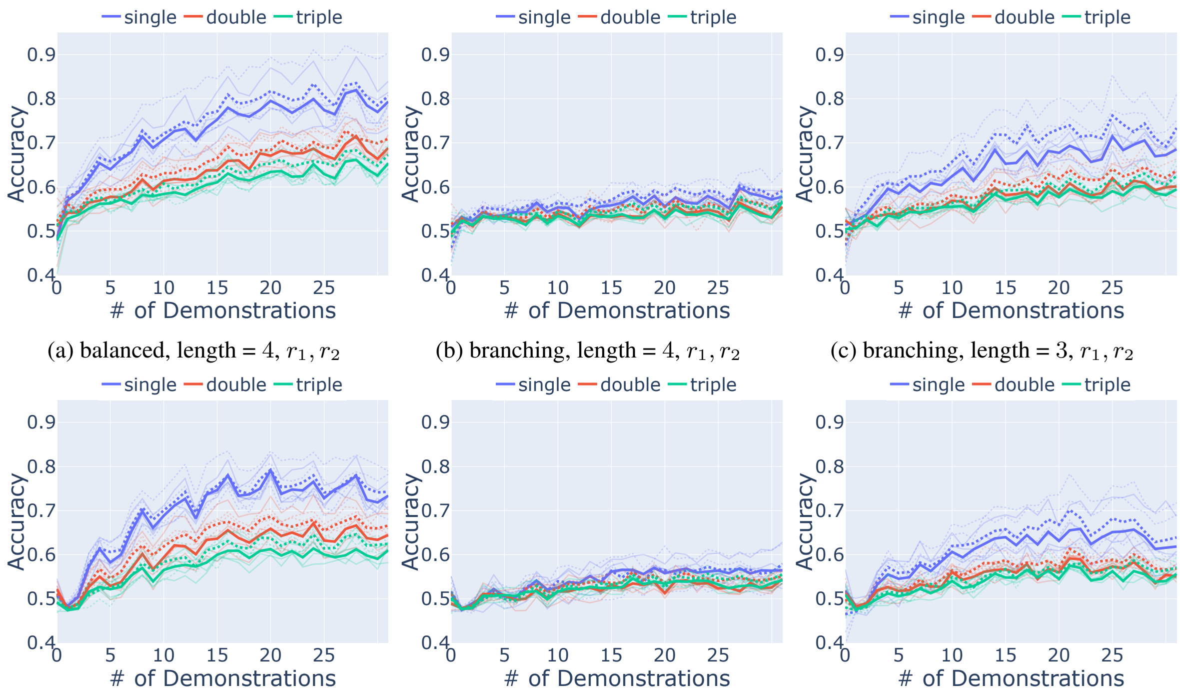
For each step i, we represent the step in the format $a^{(i)} + b^{(i)} = c^{(i)}$, where $a^{(i)}, b^{(i)}$ and $c^{(i)}$ are sequences of n tokens, each of which is in $[0, 9]$, representing a number from the lowest digit to the highest digit. $a^{(0)}$ and $b^{(0)}$ represent two randomly drawn numbers and $c^{(0)}$ is all zero. At each step $i > 0$, most of the digit in $a^{(i)}, b^{(i)}, c^{(i)}$ is the same as the previous step. For $a^{(i)}$ and $b^{(i)}$, we only update the ith digit by setting $a_i^{(i)} = 0$ and $b_i^{(i)} = 0$. As for $c^{(i)}$, it serves as a buffer for both the answer and the carry. We update it based on $s^{(i)} = a_i^{(i-1)} + b_i^{(i-1)} + c_i^{(i-1)}$, the sum of the digits at i. We set $c_i^{(i)} = s^{(i)} \bmod 10$, $c_{i+1}^{(i)} = \lfloor s^{(i)}/10 \rfloor$. We use colons as the separator and concatenate these steps as a single sequence. When testing a model's intuition, we let the model generate the continuation for $a^{(0)} + b^{(0)} = c^{(0)}; a^{(n)} + b^{(n)} =$. Note that $a^{(n)} = b^{(n)} = 0$, so the model needs to have the *intuition* to generate the answer correctly. We provide examples in Table 3.



(a) balanced, length = 4, y_1, y_2 (b) branching, length = 4, y_1, y_2 (c) branching, length = 3, y_1, y_2

(d) balanced, length = 3, y_3, y_4 (e) branching, length = 4, y_3, y_4 (f) branching, length = 3, y_3, y_4

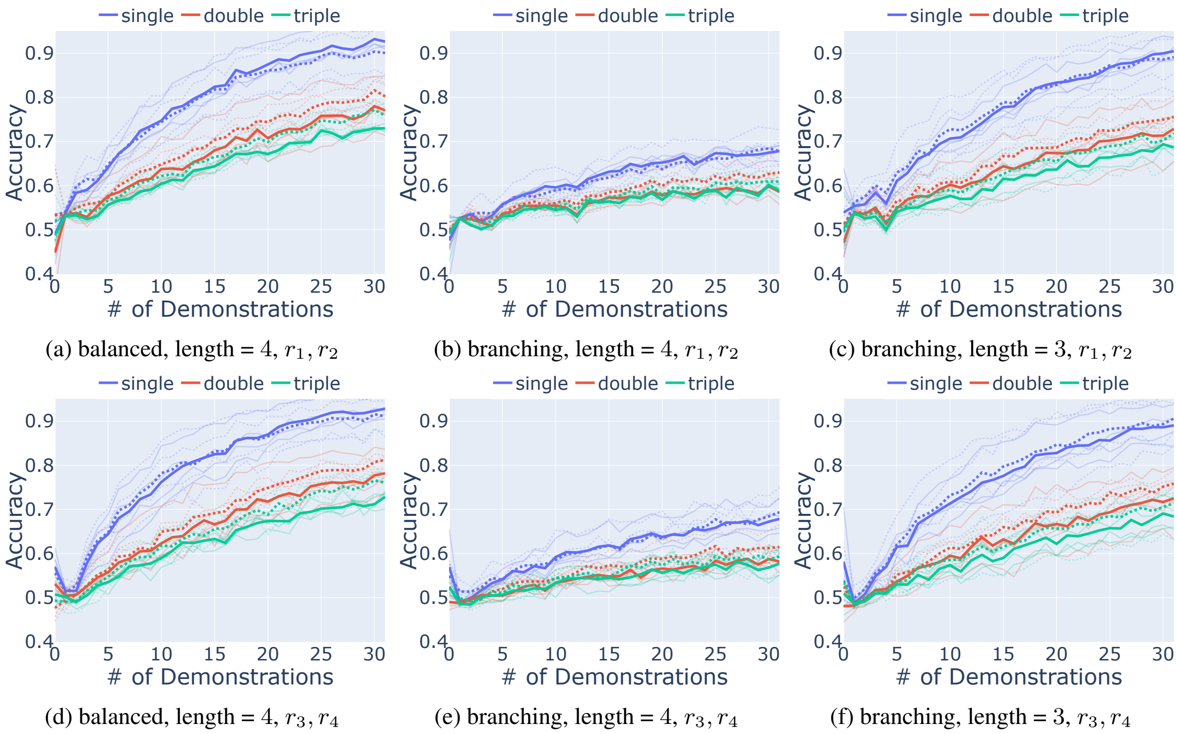
Figure 5: In-context learning accuracy with Calcutec when using different verbalizers (y_1, y_2 or y_3, y_4) and input lengths (3 or 4). The dotted lines represent the performance of *unseen combinations* described in §6.1.2, while the different colors represent the number of formulas each class (v_+ or v_-) is associated to. The main lines represent the average accuracy of 5 tasks. We plot the performance of each task in lighter colors.



(a) balanced, length = 4, r_1, r_2 (b) branching, length = 4, r_1, r_2 (c) branching, length = 3, r_1, r_2

(d) balanced, length = 4, r_3, r_4 (e) branching, length = 4, r_3, r_4 (f) branching, length = 3, r_3, r_4

Figure 6: In-context learning accuracy with Calcutec when no steps are dropped ($p_{skip} = 0$).



(a) balanced, length = 4, r_1, r_2
(b) branching, length = 4, r_1, r_2
(c) branching, length = 3, r_1, r_2
(d) balanced, length = 4, r_3, r_4
(e) branching, length = 4, r_3, r_4
(f) branching, length = 3, r_3, r_4

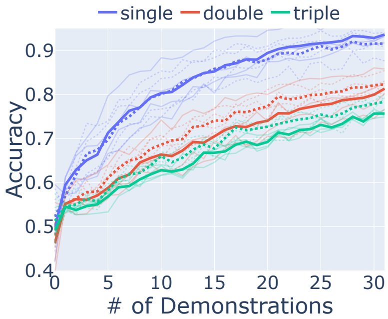
Figure 7: In-context learning accuracy with Calcutec without rewriting the first step to include contain the premise of the proof.

	Branching				Balanced			
	r_1, r_2		r_3, r_4		r_1, r_2		r_3, r_4	
Task	ICL	CoT	ICL	CoT	ICL	CoT	ICL	CoT
Single	57.1	91.7	55.6	92.0	68.5	89.8	64.9	90.3
Double	53.5	76.3	51.1	77.1	58.5	76.1	56.2	75.8
Triple	53.0	73.0	51.7	73.4	57.0	68.2	54.2	67.0

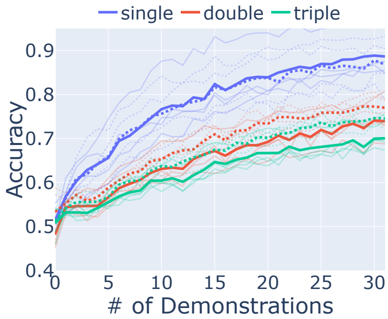
Table 5: The 4-shot accuracy of in-context learning (ICL) versus chain-of-thoughts (CoT).

	branching						balance					
	r_1, r_2			r_3, r_4			r_1, r_2			r_3, r_4		
#-shot	2	4	6	2	4	6	2	4	6	2	4	6
single	49.1	89.5	84.0	59.5	92.0	86.9	58.5	86.2	85.5	50.3	90.3	89.9
double	47.8	71.4	75.6	53.1	77.1	86.1	49.1	70.4	69.0	50.5	75.8	79.4
triple	46.7	65.7	70.7	50.6	73.4	79.4	46.0	60.2	61.4	49.8	67.0	70.4

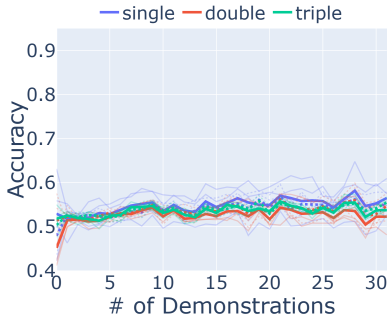
Table 6: The CoT performance with 2, 4, or 6 examples.



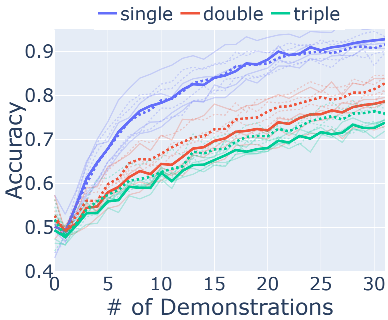
(a) balanced, r_1, r_2, 25000 examples



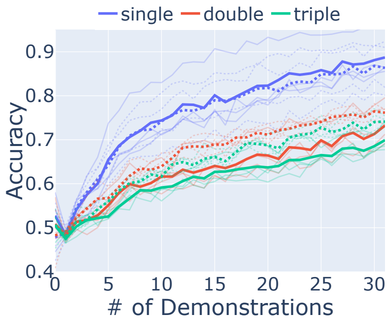
(b) balanced, r_1, r_2, 10000 examples



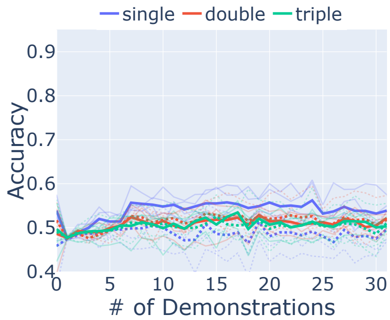
(c) balanced, r_1, r_2, 5000 examples



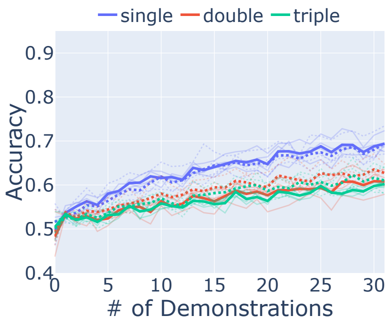
(d) balanced, r_3, r_4, 25000 examples



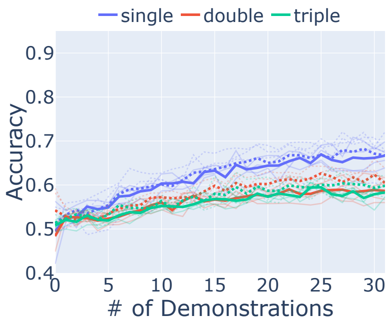
(e) balanced, r_3, r_4, 10000 examples



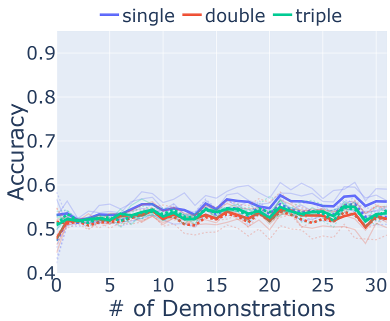
(f) balanced, r_3, r_4, 5000 examples



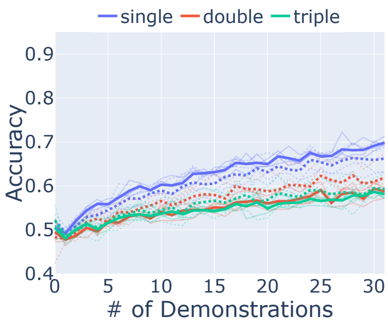
(g) branching, r_1, r_2, 25000 examples



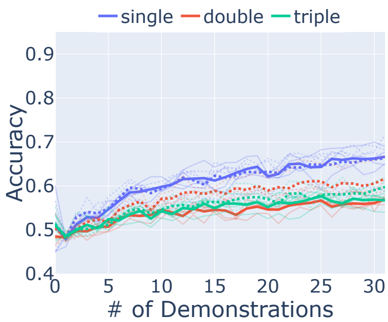
(h) branching, r_1, r_2, 10000 examples



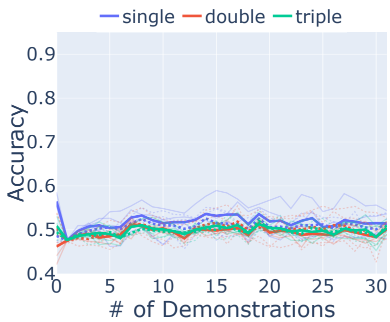
(i) branching, r_1, r_2, 5000 examples



(j) branching, r_3, r_4, 25000 examples

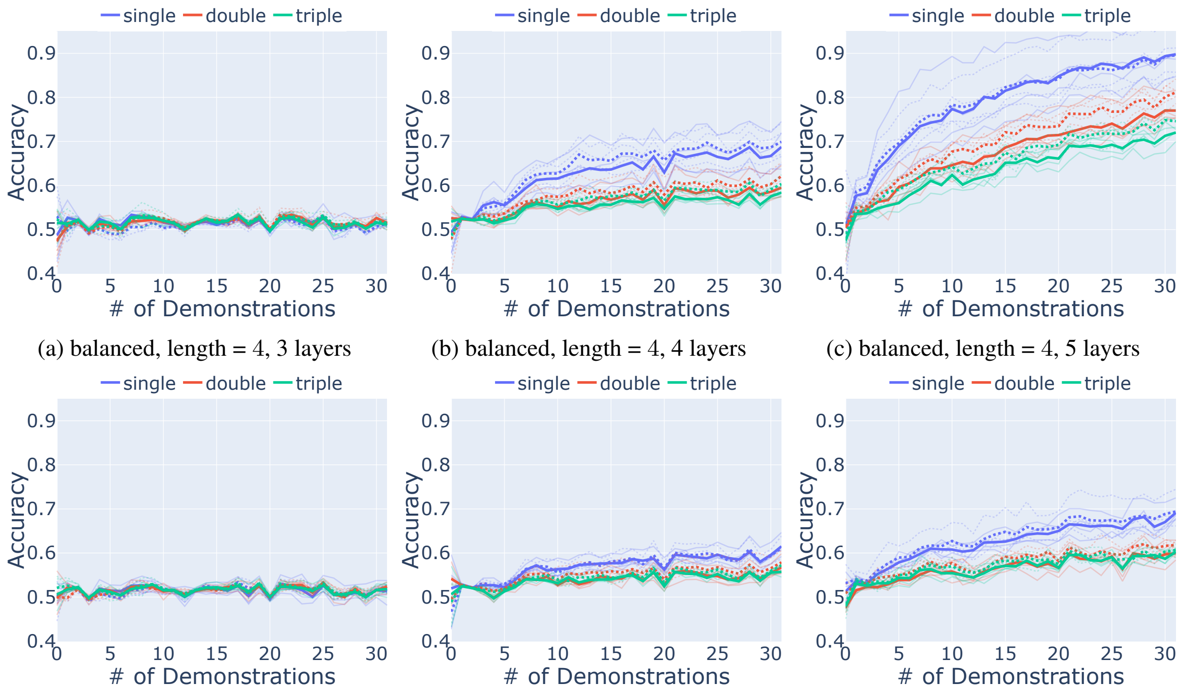


(k) branching, r_3, r_4, 10000 examples



(l) branching, r_3, r_4, 5000 examples

Figure 8: In-context learning accuracy with different sizes of



(a) balanced, length = 4, 3 layers (b) balanced, length = 4, 4 layers (c) balanced, length = 4, 5 layers

(d) branching, length = 4, 3 layers (e) branching, length = 4, 4 layers (f) branching, length = 4, 5 layers

Figure 9: The in-context learning performance when using models with different model depths.

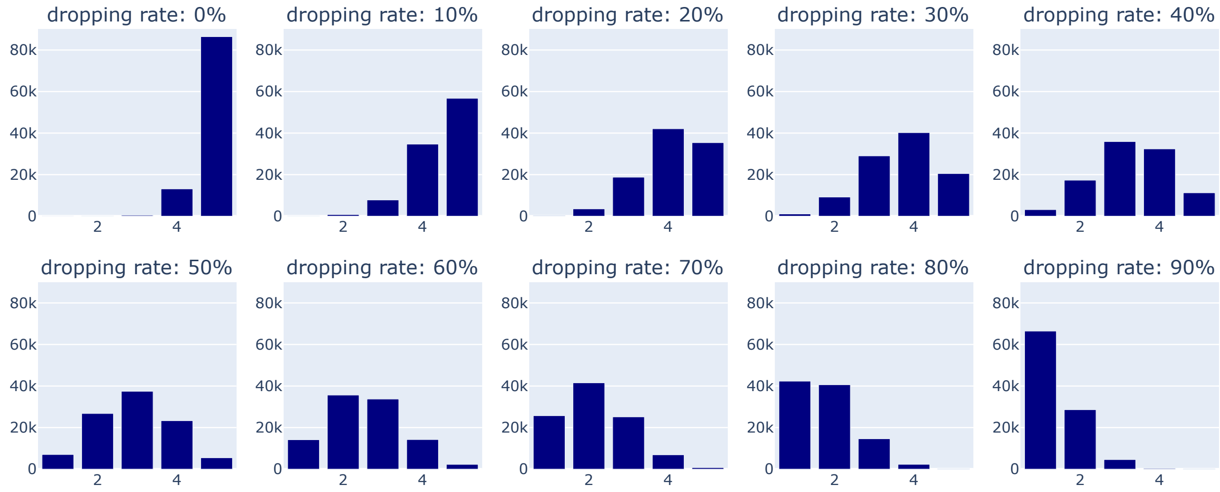
K Digit Addition with Noisy Training data

To study the effect of noises, we experiment with noisy training data for the digit addition task. In this setup, at step i, we mutate the sum digit s_i at a probability. (Please refer to J for the definition of the symbols.) The mutated digit is passed to the next step and thus cause the final result to be incorrect.

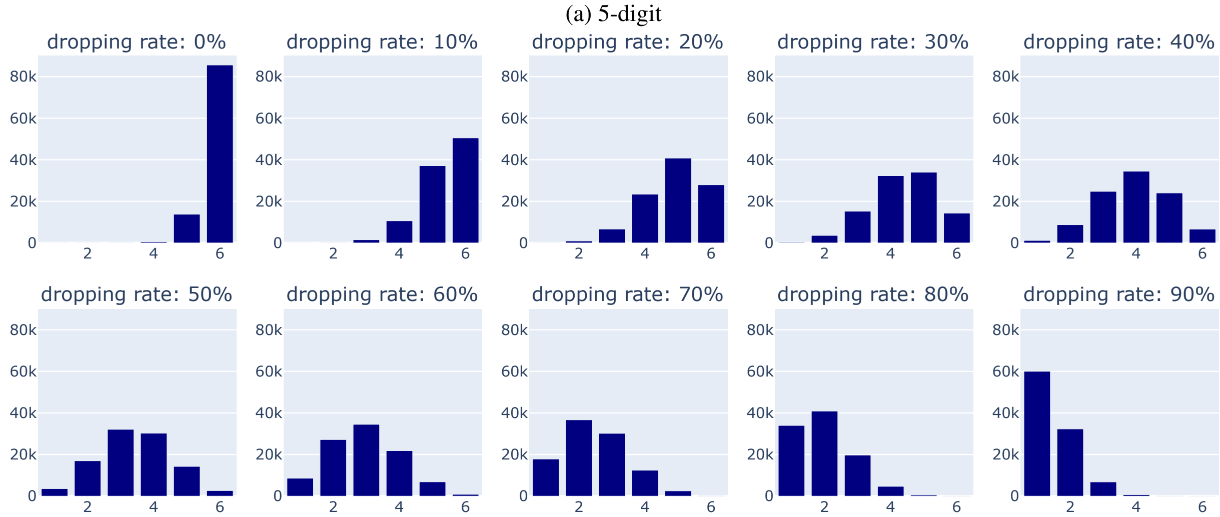
We plot the result in Figure 12. Surprisingly, the models trained with the noisy data can still achieve a 100% EM score. It suggests that the noisy training data used in practice may not prevent the model from learning the interrelation between concepts.

L Dataset License

- SST-2: MIT
- MR: unavailable
- CR: unavailable
- Subj: unavailable

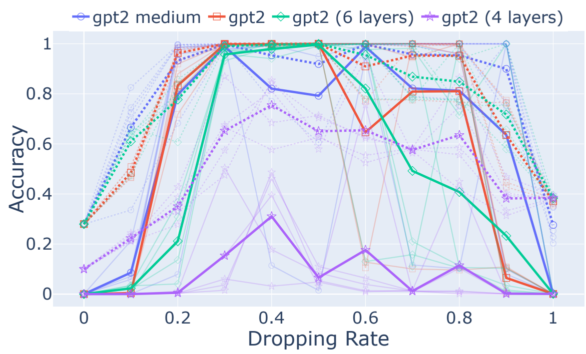


(a) 5-digit

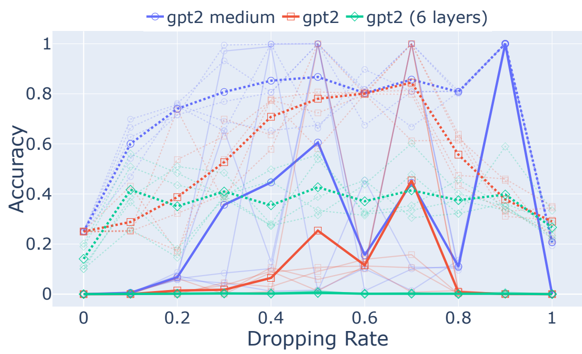


(b) 6-digit

Figure 10: The distribution of the number of reasoning steps in the dataset when some of them are dropped at different probability. Each number is the average over 5 datasets generated with different random seeds.



(a) 5 digits



(b) 6 digits

Figure 11: The accuracy of the models for the addition tasks. The x-axis represents the probability at which we drop each reasoning step in the training data independently. The solid line represents the ratio of testing samples where the model can output the exact answer, while the dashed line represents the character-level accuracy. The results are the average of 5 random seeds.

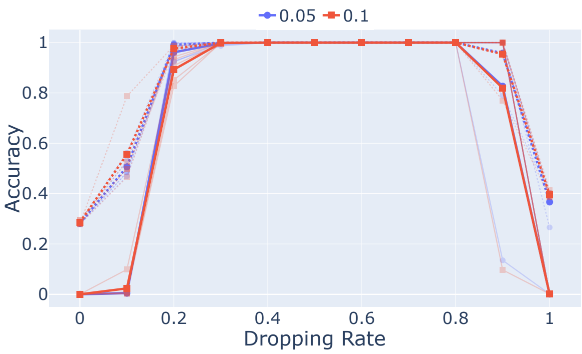


Figure 12: The accuracy of the models for the addition tasks when trained with noisy data.

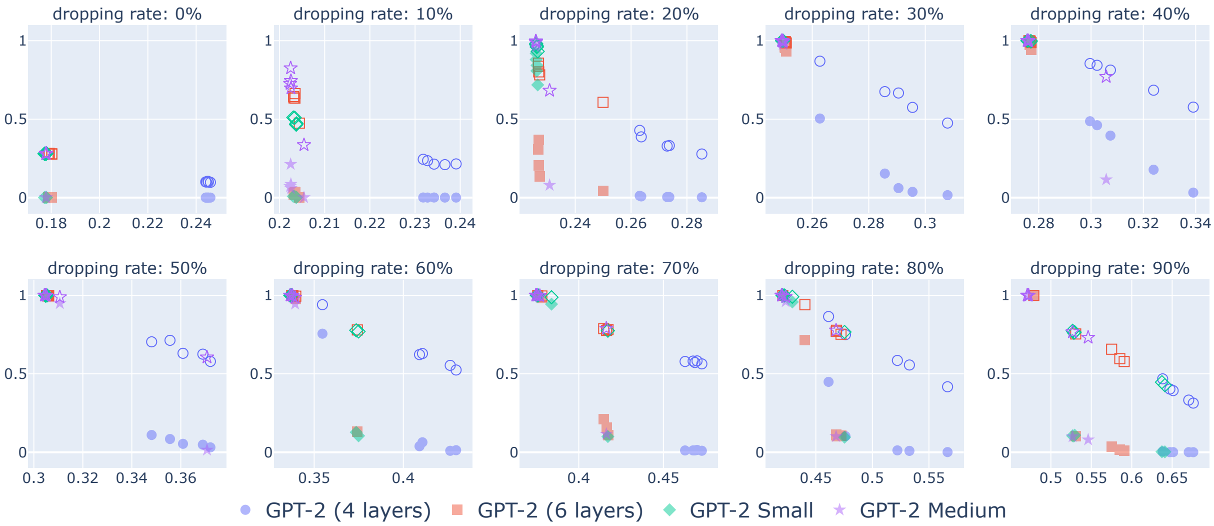


Figure 13: The exact accuracy (y-axis, solid points) and digit-level accuracy (y-axis, hollow points) versus validation loss (x-axis) for a 5-digit addition task.

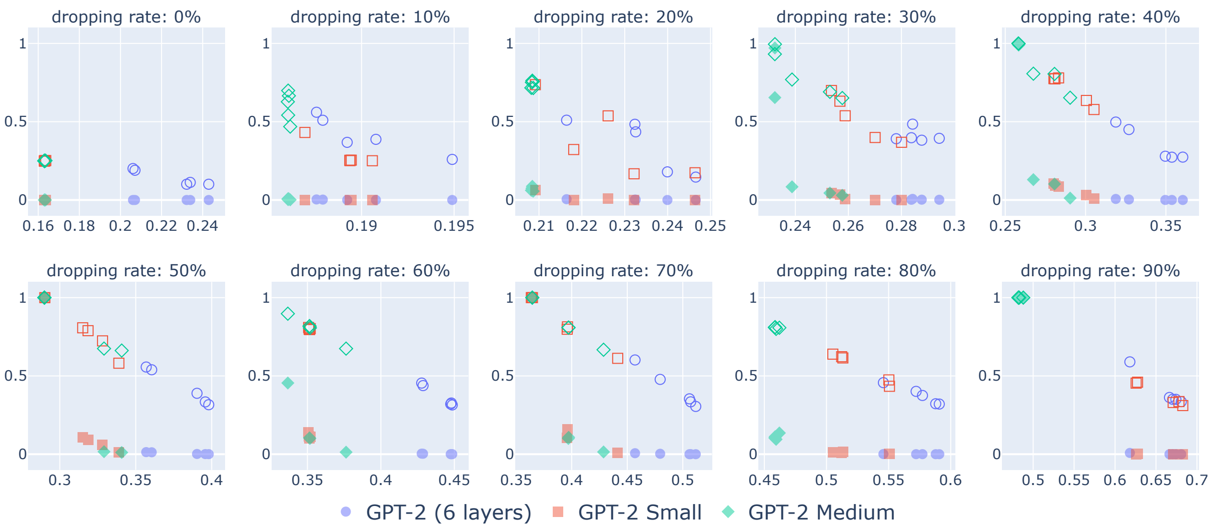


Figure 14: The exact accuracy (y-axis, solid points) and digit-level accuracy (y-axis, hollow points) versus validation loss (x-axis) for a 6-digit addition task.