Scaling Down Transformers: Investigating Emergent Phenomena in Tiny Models

Abhas Kumar Sinha
abhassriva@gmail.com

Abstract

In this paper, we investigate the emergent learning capabilities of autoregressive models, particularly Generative Pre-trained Transformers (GPTs), with a focus on how these models learn and generalize from sparse datasets. Our findings show that increasing the number of training epochs enhances emergent abilities, where different loss reductions correlate with distinct task distributions in the pretraining dataset. We demonstrate how GPT models decompose sparse data into latent patterns, allowing them to learn new distribution information and extrapolate these patterns to improve cross-task predictions. For instance, the model achieves nearly 65% accuracy with just 50 samples (out of 1680 possible samples) when encoding two synthetic languages, and reaches accuracies of 93.5%, 99.75%, and 99.25% with 100, 200, and 300 samples (out of 11880 possible samples), respectively, from a larger dataset. Additionally, we highlight the influence of prompt design, dataset characteristics, and latent variables on pattern recognition and task performance. These results advance the understanding of large language models' emergent learning abilities, offering insights into their generalization power across diverse tasks.

1 Introduction

Artificial neural networks with *attention mechanisms*—particularly transformer-based architectures have transformed natural language processing (NLP) in recent years. The paradigm of *pre-training* has yielded the most significant advances in this discipline. In short, models are trained on large corpora to predict the next word in a sequence, usually in an *autoregressive fashion*—that is, anticipating the next token based on the ones that came before it [14, 17].

The emergence of abilities in neural networks that were not explicitly taught during training has been one of the most fascinating results of pre-training. These **emergent abilities** refer to the model's capacity to generalize *beyond its training data* [18, 19, 20, 21], achieving **SoTA performance** [22, 23, 21] in tasks it wasn't directly trained for. With minimal fine-tuning or task-specific training, pre-trained models have proven their ability to perform a wide range of tasks, revealing intriguing properties like knowledge transfer, *Few-Shot Learning (FSL)*, *Zero-Shot Learning (ZSL)*, and the increasingly studied paradigm of *In-Context Learning (ICL)* [26, 7].

Even while the use of emergent abilities—such as ICL, fine-tuning, and others—has increased drastically, little has been discovered about the underlying reasons of these abilities' development. Specifically, questions remain regarding the prerequisites for such abilities to manifest, how they can be optimized, and their relationship to factors such as model size or the type of training data. Much of the current research focuses on large language models (LLMs) and vast datasets, with ongoing debates surrounding the nature of "emergent abilities." Are these phenomena genuine, or are they just an effect of non-linear error scaling, which becomes apparent up when models are scaled above particular accuracy thresholds? [16, 21] In addition, it's unclear if LLMs may generate data that is truly "out-of-distribution" or what external factors—be it the dataset itself or the specific prompt—most strongly influence the emergence of these abilities [25, 16]. Understanding these dynamics is crucial as researchers seek to optimize model performance and push the boundaries of what pre-trained models can achieve.

In this paper, we demonstrate that autoregressive models, like GPTs, function more as *"pattern completion models"* rather than mere *word predictors*. These models possess the ability to recognize underlying *structures* in the data, enabling them to generalize beyond in-distribution examples. By identifying **latent patterns**, they can efficiently learn from *sparse* data, making it possible to extrapolate these patterns to generate coherent outputs in new data distributions. This capacity for *pattern recognition* and *completion* highlights the emergent learning behavior of these models, particularly in tasks involving *minimal training data*. We empirically demonstrate that, in contrast to certain beliefs, GPT models are *unable* to generate data that is truly out-of-distribution. Nevertheless, we will also show that these models can achieve exceptional task generalization with a minimal quantity of prior data and a small number of samples. We want to add important insights to the current conversation on emergent skills and LLM operational frameworks by illuminating these phenomena.

In summary, the key contributions of this work are as follows:

- **Impact of Epochs on Emergent Abilities:** This study demonstrates how increasing the number of training epochs enhances emergent abilities. Different loss drops correspond to distinct task distributions in the pretraining dataset, leading to improved abilities with minimal examples for certain distributions.
- **Latent Variable Learning in Sparse Data:** We show how large language models (LLMs) learn distributions from sparse data by leveraging different latent variables. These variables decompose sparse data into underlying patterns, enabling the models to extract new distribution information.
- **Extrapolation of Patterns for Cross-Task Predictions:** The research explores how LLMs extrapolate learned patterns to generate more accurate predictions across various tasks, demonstrated through the *triangle translation problem* in the results section 3.1.3.
- **Effects of Prompt, Dataset, and Latent Variables on Pattern Recognition:** The study investigates the role of prompting, dataset characteristics, and latent variables in the pattern recognition process of LLMs, revealing their impact on emergent learning abilities.

2 A minimal setup for *Emergent Learning*

Emergent Learning refers to the phenomenon where a Large Language Model (LLM) demonstrates capabilities it has not been explicitly trained for or where specific task examples are sparse in the dataset compared to the dataset as a whole [18, 19, 20, 21]. In this paper, we examine a particular subset of emergent learning to explore how it manifests in a small language model and evolves over time as the number of parameters and the size of the dataset increase. Specifically, we focus on the *synthetic translation* or *encoding problem*, where a basic GPT model is trained to approximate *discrete input-output* value functions. For example, in our first experiment (described in the Results section), we consider three sets of discrete tokens, A, B, and C, with $|A| = |B| = |C| = 8$. The model is trained with sufficient samples of the functions $f : A \to B$, $g : B \to C$, and a third function $h : A \to C$. The goal is to generalize h with some small number of samples - required for the model to effectively approximate the function h, given an adequate number of samples of the functions f and g, thereby enabling the LLM to generalize and *encode* a fixed-length sentence task that it has minimally been trained to.

2.1 LLM Training analysis for emergent ability

Proposition 1. *Given a model trained on tasks T_1 and T_2, where the number of samples for task T_1 (N_{T_1}) is significantly larger than that for task T_2 (N_{T_2}), the model initially converges primarily towards minimizing the loss for task T_1 due to the imbalance in sample sizes. Over time, the loss contributions from task T_2 become more significant, leading the model to a state where both losses are minimized.*

Proof. Let the total loss $\mathcal{L}(T_1, T_2)$ be defined as the sum of the losses from tasks T_1 and T_2:

$$\mathcal{L}(T_1, T_2) = \frac{1}{N_{T_1}} \sum_{i=1}^{N_{T_1}} \mathcal{L}_{T_1}(x_i) + \frac{1}{N_{T_2}} \sum_{j=1}^{N_{T_2}} \mathcal{L}_{T_2}(x_j)$$

where $\mathcal{L}_{T_1}(x_i)$ and $\mathcal{L}_{T_2}(x_j)$ are the individual loss functions for tasks T_1 and T_2, respectively.

During gradient descent, the model updates its weights w at each iteration t according to the rule:

$$w_{t+1} = w_t - \eta \cdot \nabla_w \mathcal{L}(T_1, T_2)$$

where η is the learning rate, and $\nabla_w \mathcal{L}(T_1, T_2)$ is the gradient of the total loss with respect to the weights. The gradient of the total loss is the sum of the gradients from both tasks:

$$\nabla_w \mathcal{L}(T_1, T_2) = \frac{1}{N_{T_1}} \sum_{i=1}^{N_{T_1}} \nabla_w \mathcal{L}_{T_1}(x_i) + \frac{1}{N_{T_2}} \sum_{j=1}^{N_{T_2}} \nabla_w \mathcal{L}_{T_2}(x_j)$$

Since $N_{T_1} \gg N_{T_2}$, the term associated with task T_1, $\frac{1}{N_{T_1}} \sum_{i=1}^{N_{T_1}} \nabla_w \mathcal{L}_{T_1}(x_i)$, initially dominates the gradient update, meaning that the model first focuses on minimizing the loss for task T_1.

As training progresses and $\mathcal{L}_{T_1}$ decreases, its gradient $\nabla_w \mathcal{L}_{T_1}$ diminishes, slowing down the rate at which task T_1's loss is minimized. At this point, the gradient contribution from task T_2 becomes more significant.

The balance point, where both losses are approximately equal, occurs when:

$$\frac{1}{N_{T_1}} \sum_{i=1}^{N_{T_1}} \mathcal{L}_{T_1}(x_i) \approx \frac{1}{N_{T_2}} \sum_{j=1}^{N_{T_2}} \mathcal{L}_{T_2}(x_j)$$

Thus, the total loss $\mathcal{L}(T_1, T_2)$ becomes balanced, and the model converges to both of the tasks, completing the proof. □

We continue to discuss the impact of the above proof in the *Results* section of the paper - showing a little bit of example in the pre-training data (or fine-tuning) could have very significant impacts in the generalization abilities of the casual autoregressive models. This is sufficient for the training section, we now look forward to latent variable probability modeling of the language model in the next section.

2.2 Latent Variable Space for Patterns

Assumption 1. *A trained causal GPT language model behaves as a probabilistic model with a given* ***context*** *θ, such that its output is defined as $y_t = \arg\max_{y_t}$ LLM$(y_t|\theta, y_{1:t-1})$. Here, θ belongs to a discrete set of* ***contexts****, which we define as recurring patterns found in the examples within the pre-training dataset.*

Note: *It is important to emphasize that no search methods are used in any of the models discussed here.*

In this assumption, we introduce the notion of an additional probabilistic latent variable, consistent with prior work where this variable is often referred to as either *context* or *message* [1, 2, 3, 4]. However, we utilize it in a *different* sense, referring specifically to recurring patterns, such as repetitions or structured patterns within a given subset of the pre-training dataset. Thus, when we refer to a *context latent variable*, we imply that an LLM could rely on **one or more** sets of patterns at any given time, depending on the prompt provided. The figure above is an example of such a latent variable. See 1

Proposition 2. *An increase in the number of pre-training examples for each θ_i (or, alternatively, a decrease in the probability $\Pr(x_{1:t-1}|\theta_i)$), or a higher probability for a specific y_t token associated with a particular θ_i, indicates a context pattern within θ_i. This will subsequently dominate the output of a pre-trained model for that specific θ_i when generating an output token for a given t.*

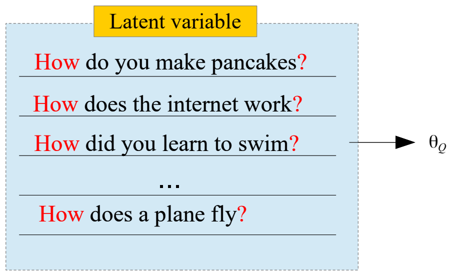

Latent variable
How do you make pancakes?
How does the internet work?
How did you learn to swim?
...
How does a plane fly?
θ_Q

Figure 1: Example of such latent variable pattern - θ_Q.

Proof. We start by defining a function for a pre-trained autoregressive causal GPT, represented as $y_t = \arg\max_{y_t} \Pr(y_t|x_{1:t-1})$, where $x_{1:t-1}$ denotes the input tokens from positions 1 to $t-1$ and y_t is the output.

$$\begin{aligned} y_t &= \arg\max_{y_t} \Pr(y_t|x_{1:t-1}) \\ &= \arg\max_{y_t} \sum_{i=1}^{n} \Pr(y_t|\theta_i, x_{1:t-1}) \\ &= \arg\max_{y_t} \sum_{i=1}^{n} \frac{\Pr(\theta_i, x_{1:t-1}|y_t)\Pr(y_t)}{\Pr(\theta_i, x_{1:t-1})} \\ &= \arg\max_{y_t} \sum_{i=1}^{n} \frac{\Pr(\theta_i|y_t)\Pr(x_{1:t-1}|y_t)\Pr(y_t)}{\Pr(\theta_i, x_{1:t-1})} \\ &= \arg\max_{y_t} \sum_{i=1}^{n} D(\theta_i, y_t, x_{1:t-1})\Pr(x_{1:t-1}, y_t) \end{aligned} \tag{1}$$

Here, $D(\theta_i, y_t, x_{1:t-1})$ is defined as $D(\theta_i, y_t, x_{1:t-1}) \triangleq \frac{\Pr(\theta_i|y_t)}{\Pr(\theta_i, x_{1:t-1})}$. It is important to note that as $D(\theta_i, y_t, x_{1:t-1})$ increases with $\Pr(\theta_i, x_{1:t-1})$, this implies a greater number of possible $x_{1:t-1}$ for each θ_i along with an increase in the probability of y_t for a given context pattern θ_i. □

Note: The final probability of the token y_t will depend on multiple context patterns that contribute to the summation, weighing the probabilities of $x_{1:t-1}$ and y against those in the pre-training data.

Thus, we conclude that the final output of a pre-trained language model is influenced by: (1) the number of training samples for a particular context pattern, (2) the likelihood that the output is part of a common vocabulary associated with a pattern in the pretraining data, and (3) the values of both (1) and (2), weighted to align with the likelihood derived from the pre-training data (indicating a *prompt*).

Proposition 3. *A task solver represented by a probability function* $y_t = \arg\max_{y_t} f(y_t|x_{1:t-1})$ *can be effectively mimicked by a well-trained autoregressive model, expressed as* $y_t = \arg\max_{y_t} f(y_t|\theta^*, x_{1:t-1})$. *This is achieved by accurately identifying the optimal context pattern* θ^* *from the input prompt* $x_{1:t-1}$ *and by increasing the number of examples in the pre-training dataset corresponding to* θ^*.

Proof. Starting with the equivalence $y_t = \arg\max_{y_t} \Pr(y_t|x_{1:t-1}) = \arg\max_{y_t} f(y_t|x_{1:t-1})$, we have:

$$
\begin{aligned}
&\Pr(y_t|x_{1:t-1}) - \sum_{i=1}^{n} \Pr(y_t|\theta_i, x_{1:t-1}) = 0 \\
&\Pr(y_t|x_{1:t-1}) - \Pr(y_t|\theta^*, x_{1:t-1}) = - \sum_{i=1:\theta_i \neq \theta^*}^{n} \Pr(y_t|\theta_i, x_{1:t-1}) \\
&|| \Pr(y_t|x_{1:t-1}) - \Pr(y_t|\theta^*, x_{1:t-1})|| \leq \epsilon(y_t, \theta^*, x_{1:t-1})
\end{aligned}
\tag{2}
$$

Here, $\epsilon(y_t, \theta^*, x_{1:t-1})$ is defined as $\epsilon(y_t, \theta^*, x_{1:t-1}) \triangleq \sum_{i=1:\theta_i \neq \theta^*} \Pr(y_t|\theta_i, x_{1:t-1})$. As the number of examples for θ^* and the prompt $x_{1:t-1}$ increases, we find that $\epsilon(y_t, \theta^*, x_{1:t-1}) \rightarrow 0$ as $\sum_{i=1:\theta_i \neq \theta^*}^{n} D(\theta_i, y_t, x_{1:t-1}) \Pr(x_{1:t-1}, y_t) \rightarrow 0$. □

Assumption 2. *A Causal GPT model learns latent pattern variables and their compositions from the dataset, enabling it to generalize from sparse distributions. It decomposes new or sparse distributions into latent variables $\theta_{1:n}$, learning only the necessary patterns from the new data, and thus performs better than a freshly trained model on the same task.*

This assumption is demonstrated in Figure 3 and is experimented with in Section 3.1 of the results. The model is very good with re-using different patterns and with just tiny hints in the dataset, it generalizes the whole distribution with sparse example present in the dataset.

2.3 Model Embeddings for cross translation tasks

In this context, we focus on a specific case of the encoding problem, where a model can uniquely encode elements from two sets: A to C and B to C. If a given model can encode both A and B into C with minimal fine-tuning or a small sample required for one embedding to adapt to the other with only slight perturbations, we highlight some theoretical relationships concerning the embeddings of the tokens from sets A, B, and C. We consider a simple scenario involving a single-head attention mechanism in a Linear Self-Attention (LSA) transformer layer, which serves as a simplified approximation of the Softmax Self-Attention (SSA) mechanism used in standard transformers, in line with current literature [5, 8, 9, 10]. This is significant because even a tiny deviation due to perturbation can lead to drastic changes in practical outcomes.

Lemma 1. *Let $z_j \leftarrow z_j PV \sum_{i=1}^{n} z_j f(z_i, z_j)$, where $f(,)$ is perturbed by a small ϵ, i.e., $f(z_i, z_j) \rightarrow f(z_i, z_j) + \epsilon$. Then, the output z_j remains largely unaffected, with the difference bounded by $\mathcal{O}(\epsilon)$.*

Proof. The perturbed expression becomes

$$z'_j \leftarrow z_j PV \sum_{i=1}^{n} z_j \left(f(z_i, z_j) + \epsilon\right) = z_j PV \sum_{i=1}^{n} z_j f(z_i, z_j) + \epsilon z_j PV \sum_{i=1}^{n} z_j.$$

By Taylor expansion of $f(z_i, z_j) + \epsilon$, we get

$$z'_j = z_j PV \sum_{i=1}^{n} z_j f(z_i, z_j) + \epsilon z_j PV \sum_{i=1}^{n} z_j + \mathcal{O}(\epsilon^2).$$

Since ϵ is small, the perturbation affects z_j only linearly, with the difference from the original z_j being of order $\mathcal{O}(\epsilon)$, and higher-order terms are negligible. □

Lemma 2. *Consider a pre-trained transformer that can generalize the mappings $A \rightarrow B$, $B \rightarrow C$, and $A \rightarrow C$, such that both A and B can be translated into C. Let the input for the mapping $A \rightarrow C$ be represented by the embeddings x_{AC} and similarly let the embeddings for $B \rightarrow C$ be denoted as x_{BC}. If we assume that $x_{AC} + \delta = x_{BC}$ yields the same correct output for C in the model, and that the model is a single-layer, single-head LSA transformer, then there exists a relationship among δ_{ij}, x_{AC}, and x_{BC}.*

Proof.

$$z_j \leftarrow z_j PV \sum_{i=1}^{n} z_j (z_i^T K^T Q z_j)$$
$$\Rightarrow f(z_i, z_j) = f(z_i + \delta_i + z_j + \delta_j)$$

Since both models produce similar outputs, e.g. same tokens from the set C. Here we ignore the terms that are less significant and consider only till attention layer outputs.

$$\begin{aligned} z_i^T K^T Q z_j &= z_i^T K^T Q z_j + z_i^T K^T Q \delta_j + \delta_i^T K^T Q z_j + \delta_i^T K^T Q \delta_j \\ 0 &= z_i^T K^T Q \delta_j + \delta_i^T K^T Q z_j + \delta_i^T K^T Q \delta_j \\ &\Rightarrow z_i^T K^T Q \delta_j + \delta_i^T K^T Q z_j + \delta_i^T K^T Q \delta_j = 0 \end{aligned} \tag{3}$$

□

2.4 Generalization ability baseline

Proposition 4. *The baseline for emergent ability is primarily determined by* ***memorization****; that is, it reflects the ability of transformer models to* ***recall*** *specific examples presented during training from the entire set of possible samples, while random outputs for the rest of the test set examples. Any performance exceeding this baseline above a certain threshold contributes to the model's generalization ability in emergent learning scenarios.*

Proof. Let $|N|$ represent the total number of possible instances, and let x denote the samples used during the pre-training phase, where $x \ll |N|$. A fully trained model that relies on memorization is subsequently tested with n samples.

Assuming the n test samples may include repetitions, we consider that the pre-training samples x contain no repetitions for simplicity.

The probability of the memorization model correctly predicting the test set samples is given by $n \cdot \frac{x}{|N|}$. For simplicity, let us define a metric function $L : N \times N \to [0, 1]$, leading to the expected loss:

$$\mathbb{E}(L) = \frac{x}{|N|}.$$

By applying Hoeffding's inequality, we have:

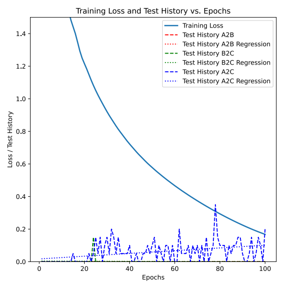
$$\begin{aligned} \mathbb{P}\left(\left|S_n - n \cdot \frac{x}{|N|}\right| \geq t\right) &\leq 2 \exp \frac{-2t^2}{n} \\ \mathbb{P}\left(\hat{\mu} - \mathbb{E}[\hat{\mu}] \geq t\right) &\leq 2 \exp\left(-2nt^2\right) \end{aligned} \tag{4}$$

Here, t refers to the deviation in the loss of the test set results and the expected results as the probability of random hits if the model doesn't generalize and outputs random variables for the examples it hasn't been trained to work on. Here, since our accuracy is being measured in *average* of test samples - we consider the mean $\hat{\mu}$ version of the equations. □

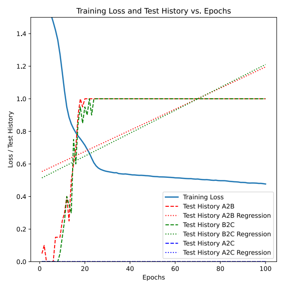
3 Results

3.1 Tringle translation problem

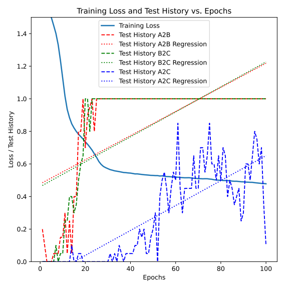
In this study, we evaluate the AR GPT model's emergent ability in a simple *triangle encoding problem* involving three sets, where $|A| = |B| = |C| = 8$ in Section 3.1.3 experiments and 12 for 3.2 experiments. Each sentence is 4 tokens long and no repetitions are allowed, resulting in 1680 possible combinations of examples for dataset possible in the Section 3.1.3 experiments and 11880 possible combinations of examples for dataset possible in the Section 3.2 experiments. The objective is to train the model to encode tokens uniquely mapped from A to B using a limited number of examples. We then introduce a few $A \to C$ encoding samples into the pre-training dataset and assess the model's generalization accuracy without extensive training on the $A \to B$ and $B \to C$ tasks.



(a) Slow to generalize; the generalization of the model is very slow and negligible.



(b) A to C encoding generalization is absolute zero as there are no A to C examples in the training data.



(c) Generalization keeps growing positively with almost 65% accuracy in the 100th epoch, despite only 50 samples.

Figure 2: Generalization performance across different conditions.

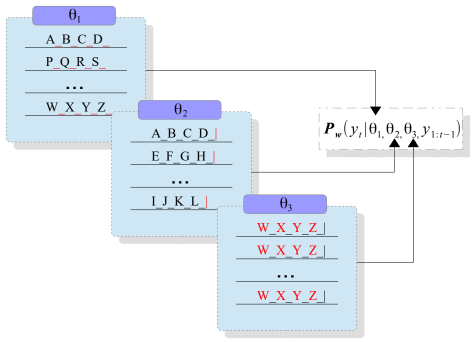


Figure 3: The above figure demonstrates how common latent variable patterns across different dataset chunks are learned by the model, with θ_1, θ_2, and θ_3 indicating three different datasets.

We first experiment with minimal $A \rightarrow C$ encodings to evaluate their sufficiency for generalization. Next, we test $A \rightarrow C$ encoding without explicit training for that task. Finally, we add several $A \rightarrow C$ samples to the pre-training dataset to compare generalization accuracy and examine whether it improves with the increased scale of pre-training examples alongside the existing $A \rightarrow B$ and $B \rightarrow C$ tasks.

Additionally, we provide an example prompt with the corresponding problem input, where each sentence consists of four tokens separated by spaces. Two special tokens are used: the *instance split* token "|" to separate example instances and the token "." to delineate input-output patterns within the problem. All the experiments in Section 3.1.3 were trained through 100 epochs and for 3.2 were 200 epochs.

Let $O = (o_1, o_2, o_3, o_4)$ be the output tokens from the model and $T = (t_1, t_2, t_3, t_4)$ be the correct output tokens. The score $\text{score}(O, T)$ evaluates the model's output O against the correct output T. It assigns a score of 1 for all four tokens matching, 0.75 for three matches, 0.5 for two matches, 0.24 for one match, and 0 for no matches, reflecting the accuracy based on the number of correctly predicted tokens in the correct order.

Next, the overall score can be calculated as:

$$\text{Total Score} = \frac{1}{\text{num\_test}} \sum_{i=1}^{\text{num\_test}} \text{score}(O_i, T_i)$$

Table 1: Comparison of Encoding Tasks with Accuracy

-	A $\rightarrow$ B	A $\rightarrow$ C	A $\rightarrow$ D	D $\rightarrow$ B*	D $\rightarrow$ C*	B $\rightarrow$ C*
# Examples in Pre-training Data	5000	4000	4500	100	200	300
Accuracy, $n = 100$ (%)	100	100	100	93.5	99.75	99.25

where O_i is the output for the i-th test case and T_i is the corresponding correct output. In your implementation, num_test $= 5$. Such metric functions that scale non-linearly ensure that no *mirage effect* creep into evaluating the emergent abilities of the models [16].

3.1.1 A to C Encoding Alone

We begin by testing the AR GPT model with 50 $A \rightarrow C$ samples to determine if the model can achieve perfect accuracy with such a limited number of examples.

After conducting the experiment, it becomes evident that the model struggles to generalize effectively from as few as 50 input-output examples related to the problem. See Figure 2(a) for the results.

3.1.2 With No A to C Samples in the Training Data

We now test the models' capabilities for *out-of-distribution* generalization by providing 2500 examples each of $A \rightarrow B$ and $B \rightarrow C$ in the training data. We aim to determine whether the models can generalize $A \rightarrow C$ encoding based solely on the previous examples.

Despite demonstrating sufficient generalization abilities for $A \rightarrow B$ and $B \rightarrow C$, the model fails to effectively generalize the $A \rightarrow C$ encoding tasks. See Figure 2(b) for the results.

3.1.3 With small samples of A to C Examples in the training data

We now test the models with the previous dataset configurations, adding 50 samples of $A \rightarrow C$ example samples into the previous dataset settings and check if the model is able to re-use previous data of different tasks and a few hints of the A to C training data to be able to generalize it with as few as samples as possible.

Now we see the model is starting to be able to generalize the required encoding function from as few as samples as possible and has been able to go over 70% of the accuracy on the test set this time. See Figure 2(c) for the results.

From Proposition 4, it is clear that the accuracy surpasses the memorization ability - which keeps on increasing with the number of epochs - indicating a reduction in both - Type - I and Type II hypothesis errors from the graph alone. The chance that the model gets 60% accuracy score by memorization is less than or equal to 7.74% for $n = 5$ test samples.

The accuracy on the $A \rightarrow C$ task increases sharply after 40th epoch when the overall loss function drops below 0.6, according to our Proposition 1 - the loss function of the small sample of out-of-distribution pre-training data of the dataset would start converging when the overall loss becomes small enough to fit the curve of the majority of the dataset distribution samples.

3.2 Testing the Impact of Pre-training Data and Sample Size on Accuracy

In this section, we introduce four languages, maintaining a similar approach and settings as in the previous section. The only difference is that the number of majority distributions and the sample examples for generalization vary. The accuracy of each model is compared and summarized in the table. See Figure 4 for the map and Table 1 for the results.

* Sparsely populated samples in the training dataset.

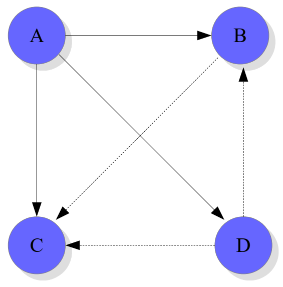


Figure 4: Results from the experiment with four hypothetical languages: A, B, C, and D. Thick lines indicate high levels of pre-training data, while dotted lines represent sparse data, demonstrating the generalization problem with varying example samples.

4 Limitations and Future work

4.1 Limitations

This study presents several limitations that should be acknowledged. Firstly, our results are derived from relatively primitive and simple tasks, which, while effective in controlled environments, do not fully encapsulate the complexities of emergent learning across a broader spectrum of applications. In particular, the findings may not adequately address tasks involving mathematical logic, such as the generalization of multiplication or division by language models, or regression tasks in domains not covered during training. The primary focus of this research has been on language translation tasks, and it remains uncertain how these findings apply to predicting or understanding real-world datasets.

Additionally, the experimental framework did not incorporate various recurrent neural network architectures, such as Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks, even though the theoretical insights could potentially extend to these models as well. Thus, the applicability of our conclusions to other time-series models and their emergent learning capabilities remains an open question.

4.2 Future Work

Despite these limitations, this paper lays a solid groundwork for future research on complex datasets and intricate tasks, exploring how In-Context Learning (ICL) may emerge in these contexts. Future investigations could aim to analyze the performance of language models on more sophisticated tasks that require a deeper understanding of context and emergent behavior.

Moreover, extending this study to include prefix language models may provide further insights into how context patterns operate within these architectures, analogous to the discussions presented here regarding causal language models. This research could contribute significantly to the theoretical foundation for various time-series models, facilitating a deeper exploration of their capabilities in handling emergent learning phenomena.

In summary, while this work contributes valuable insights into the behavior of causal language models, further exploration into more complex tasks and architectures will enhance our understanding of emergent learning and its practical implications.

5 Conclusion

In this paper, we explored the emergent abilities of large language models (LLMs), phenomena that have been previously understood primarily in the context of highly trained models. Our investigation highlights how these models can learn complex behaviors without explicit training or with only minimal exposure to random datasets. We introduced the triangle translation problem, which involves encoding tokens from language A to B and then from B to C, to evaluate the encoding capabilities of A to C with sparse training data. This unique approach ensures that each token in the synthetic languages is uniquely correlated with others, enabling a more focused analysis of emergent behaviors.

Our findings indicate that the emergent abilities of LLMs are indeed present, as evidenced by our proposed loss drop proposition 1. We observed that with loss drop, less dominant sparse distributions can become highly influential, compelling the model to converge it simultaneously with the main distribution 2(c).

Furthermore, our experiments reinforce the hypothesis that the chances of memorization are minimal. In Experiment 3.1.3, we demonstrated that the model effectively learns new distributions from the sparse data, while Experiment 3.2 showed a remarkably high accuracy, further confirming that memorization is not a factor in the model's performance. These results collectively underscore the capacity of LLMs to abstract and generalize from sparse datasets, highlighting their potential for solving complex translation tasks and contributing to our understanding of emergent learning in Large Language Models.

Acknowledgments and Disclosure of Funding

We would like to thank Google for providing free access to computational resources - GPU and TPUv2 from the Google Colab, which were instrumental in conducting the experiments for this work. We also thank the reviewers for their valuable feedback and the readers for their interest in our research.

This work did not receive any specific funding, and the authors declare no competing financial interests. Code for the repository: https://github.com/abhaskumarsinha/Scaling-Down-Transformers-Investigating-Emergent-Phenomena-in-Tiny-Models

References

[1] Xie, S.M., Raghunathan, A., Liang, P. & Ma, T. (2021) An explanation of in-context learning as implicit Bayesian inference. *arXiv preprint arXiv:2111.02080*.

[2] Wang, X., et al. (2024) Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning. In *Advances in Neural Information Processing Systems 36*.

[3] Jiang, H. (2023) A latent space theory for emergent abilities in large language models. *arXiv preprint arXiv:2304.09960*.

[4] Wies, N., Levine, Y. & Shashua, A. (2024) The learnability of in-context learning. In *Advances in Neural Information Processing Systems 36*.

[5] Ding, N., et al. (2023) CausalLM is not optimal for in-context learning. *arXiv preprint arXiv:2308.06912*.

[6] Wei, J., et al. (2023) Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

[7] Dong, Q., et al. (2022) A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

[8] Garg, S., et al. (2022) What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems 35*, pp. 30583–30598.

[9] Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A. & Vladymyrov, M. (2023) Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR.

[10] Zhang, R., Frei, S. & Bartlett, P.L. (2023) Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*.

[11] Sinha, A.K. (2024) Corpus2GPT. *GitHub Repository*. Available at: `https://github.com/abhaskumarsinha/Corpus2GPT/`.

[12] Chollet, F. (2015) Keras. *GitHub Repository*. Available at: `https://keras.io`.

[13] Abadi, M., et al. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Available at: `https://www.tensorflow.org`.

[14] Vaswani, A., et al. (2017) Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008.

[15] Radford, A., et al. (2019) Language models are unsupervised multitask learners. *OpenAI blog 1*(8), pp. 9.

[16] Schaeffer, R., Miranda, B. & Koyejo, S. (2024) Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems 36*.

[17] Guu, K., et al. (2020) Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR.

[18] Wei, J., et al. (2022) Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

[19] Ganguli, D., et al. (2022) Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*.

[20] Srivastava, A., et al. (2022) Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

[21] Brown, T.B., et al. (2020) Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

[22] Paperno, D., et al. (2016) The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

[23] Joshi, M., et al. (2017) TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

[24] Kong, L., et al. (2020) Calibrated language model fine-tuning for in-and out-of-distribution data. *arXiv preprint arXiv:2010.11506*.

[25] Liu, B., et al. (2023) How good are large language models at out-of-distribution detection? *arXiv preprint arXiv:2308.10261*.

[26] Liu, Haokun, et al. "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning." *Advances in Neural Information Processing Systems 35* (2022): 1950–1965.