Emergent Abilities in Reduced-Scale Generative Language Models

Sherin Muckatira, Vijeta Deshpande, Vladislav Lialin, Anna Rumshisky
University of Massachusetts Lowell
{sherinbojappa_muckatira, vijeta_deshpande}@student.uml.edu
{vlialin, arum}@cs.uml.edu

Abstract

Large language models can solve new tasks without task-specific fine-tuning. This ability, also known as in-context learning (ICL), is considered an emergent ability and is primarily seen in large language models with billions of parameters. This study investigates if such emergent properties are strictly tied to model size or can be demonstrated by smaller models trained on reduced-scale data. To explore this, we simplify pre-training data and pre-train 36 causal language models with parameters varying from 1 million to 165 million parameters. We show that models trained on this simplified pre-training data demonstrate enhanced zero-shot capabilities across various tasks in simplified language, achieving performance comparable to that of pre-trained models six times larger on unrestricted language. This suggests that downscaling the language allows zero-shot learning capabilities to emerge in models with limited size. Additionally, we find that these smaller models pre-trained on simplified data demonstrate a power law relationship between the evaluation loss and the three scaling factors: compute, dataset size, and model size.[1]

1 Introduction

Recent advancements in deep learning and distributed computing have enabled the pre-training of language models on a massive scale (Brown et al., 2020; Bubeck et al., 2023; Touvron et al., 2023), significantly changing the way these models are used. Large pre-trained models proved capable of solving various tasks with zero-shot or few-shot learning, eliminating the need for task-specific fine-tuning (Brown et al., 2020). This is referred to as in-context learning, an ability which allows these models to understand and solve new tasks based on the provided context. It is argued that this ability "emerges" with a dramatic increase in the size of the model (Wei et al., 2022a).

Efforts to transfer emergent abilities to small models include imitation learning, where a large language model like GPT-4 acts as a "teacher" to create synthetic datasets with additional instructions and explanations. This synthetic data is then used to train smaller "student" models (Taori et al., 2023; Peng et al., 2023; Mukherjee et al., 2023; Magister et al., 2023). Another approach is distillation where the "student" model is trained to mimic the output probabilities of the "teacher" model (Gu et al., 2023; Xu et al., 2024) .

Our work takes a different approach; our goal is to determine whether simplifying the pre-training data itself can unlock emergent language abilities in smaller models. This idea is supported by our previous work (Deshpande et al., 2023), which highlighted the effects of language simplification for smaller models when fine-tuning on downstream tasks. Prior work by Eldan and Li (2023) reports a similar trend, though their approach requires the use of larger models to produce the simplified language. We bypass this step and instead rely on naturally-occurring language restricted via vocabulary filtering.

In this study, we leverage this approach to determine whether language simplification can unlock ICL abilities in smaller language models. To do so, we pre-train 36 causal language models with sizes varying from 1M to 165M parameters, on both a simplified English dataset and a standard pre-training dataset and conduct zero-shot evaluations on different tasks. Through extensive experimentation, we show that language simplification enables ICL abilities in smaller language models on a level comparable to larger-size models pre-trained on non-simplified English corpora.

Specifically, our contributions are as follows:

[1]Code and simplified pre-training data are available at github.com/text-machine-lab/mini_gpt

Findings of the Association for Computational Linguistics: NAACL 2024, pages 1242–1257

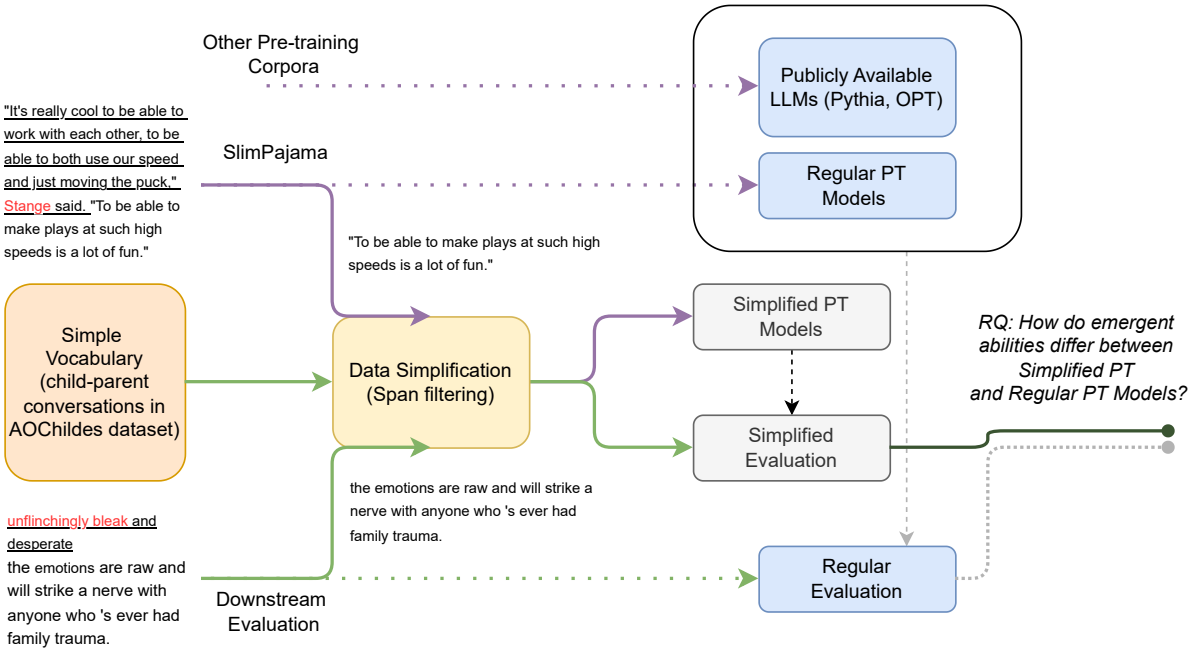


Figure 1: We filter the SlimPajama dataset by selecting spans that contain words from the AO-Childes vocabulary and removing any spans with words not in this vocabulary. We also filter examples in the downstream evaluation dataset based on the occurrence of words in the AO-Childes Vocabulary. The underlined spans are removed by filtering due to the presence of Out of Vocabulary words (Out of Vocabulary words are in red). This simplified dataset is used to pre-train simplified models, whereas regular models are trained on the standard SlimPajama dataset or on other existing pre-training corpora. We then compare whether simplified pre-trained models can perform downstream tasks in simplified language as effectively as standard pre-trained models do in the complete language.

- We demonstrate that downscaling (simplifying) the language enhances zero-shot learning capabilities in smaller-sized models.
- We show that small models trained with such simplified data demonstrate a power law relationship between evaluation loss and the three scale factors: FLOPs, Dataset Size, and Model Size.
- We release a simplified pre-training corpus obtained by filtering the existing SimPajama dataset (Soboleva et al., 2023).

Split	Percentage of tokens	Number of tokens (mil)
C4	23.86%	5258.73
GitHub	0.21%	46.10
Commoncrawl	22.12%	4875.09
StackExchange	1.33%	293.06
Wikpedia	0.08%	18.49
ArXiv	0.53%	117.66
Books	51.86%	11429.27
Total	100%	22038.41

Table 1: Data source distribution for the simplified pre-training dataset derived from SlimPajama.

2 Related Work

What is ICL? ICL is the ability of a pre-trained model to solve tasks without task specific fine-tuning (Radford et al., 2019; Brown et al., 2020; Olsson et al., 2022). Many large models have shown excellent ICL capabilities (Touvron et al., 2023; Chowdhery et al., 2022). This has shifted the research community's focus towards leveraging prompts to elicit zero-shot or few-shot responses from models. In a similar vein, the technique of chain-of-thought (CoT) reasoning, as discussed in Wei et al. (2022b), revealed that including a sequence of intermediate reasoning steps can enhance the reasoning skills of large language models. Yet, these abilities are emergent, i.e., it is primarily the larger models that exhibit them. However, recent studies question the belief that improvements in ICL result exclusively from increasing model sizes (Schaeffer et al., 2023; Du et al., 2024), suggesting that using discontinuous metrics like accuracy merely creates the illusion of emergent abilities, whereas employing continuous metrics shows gradual, predictable changes in model performance.

ICL in smaller language models. It has been shown that the emergent abilities observed in larger models can be effectively transferred to smaller models through imitation learning or behavior cloning, where a larger language model such as GPT-4 serves as the "teacher", to generate synthetic datasets with instructions and explanations which can be used to train smaller language models, referred to as "student" models (Taori et al., 2023; Peng et al., 2023; Mukherjee et al., 2023; Magister et al., 2023). This allows smaller models to leverage the capabilities of their larger counterparts. However, the primary drawback of such methods is that most of the knowledge acquired by the model is done in the pre-training stage and the student model copies the style of the teacher model but does not learn the reasoning capabilities employed by these large models (Gudibande et al., 2023).

An alternative strategy to enhance the capabilities of smaller models is through distillation from larger models, aiming to replicate the output probabilities and thus transfer the larger model's in-context learning or zero-shot abilities to their smaller counterparts (Timiryasov and Tastet, 2023; Gu et al., 2023; Xu et al., 2024). This method forfeits one of the primary benefits of smaller language models, namely their reduced computational requirements, by necessitating the training of larger models.

Prior work has also looked into pre-training small models with simplified data. For instance, Huebner et al. (2021) pre-train an encoder language model with corpus that reflects the lexical exposure of children and find that smaller models can approximate the grammatical acquisition performance of larger models. Deshpande et al. (2023) examined the effects of downscaling the modeled language during pre-training via vocabulary-based filtering, and showed that pre-training encoders as small as 1.25M parameters may demonstrate large benefits for downstream performance.

Eldan and Li (2023) have demonstrated that coherency in text generation can be achieved by pre-training on a synthesized, simplified dataset generated from GPT-4. Notably, this dataset largely comprises of stories, presenting less diversity compared to the datasets typically employed for pre-training larger models. Similarly Gunasekar et al. (2023) demonstrate improved performance in smaller models trained on a dataset combining filtered coding examples and synthetic textbook content for coding-related benchmarks. However, their approach, primarily focused on coding challenges, utilizes relatively large models and synthetic data. Their dataset filtering approach also relies on an auxiliary classifier for text exclusion.

3 Methodology

3.1 Language Simplification

We create a simpler pre-training corpus by utilizing a vocabulary derived from the AO-Childes transcripts of child-directed speech (Huebner and Willits, 2021), as done by Deshpande et al. (2023). The core of this corpus is child-directed speech, which tends towards simpler linguistic structures. The vocabulary we use comprises 21,036 unique words, reflecting the lexical range typically found in language directed at children. Filtering existing pre-training corpora with this vocabulary thus results in a simpler pre-training dataset.

3.2 Pre-training Data Collection

To obtain high quality datasets with sufficient deduplication and diversity we leverage datasets used for pre-training large language models such as the SlimPajama dataset (Soboleva et al., 2023). We begin by selecting samples from the train split of the SlimPajama pre-training corpus, then tokenize this text into distinct elements, such as words and symbols. We retain tokens that are either integers, special symbols, or belong to the AO-Childes vocabulary. This process continues until we accumulate a minimum number of tokens in a chunk. For the 22 Billion dataset the minimum number of tokens are set to 32 and for the 2.1 Billion dataset the minimum number of tokens are set to 100. We allow up to 1.5% of these tokens to be out-of-vocabulary (OOV) words to maintain a simplified vocabulary and yet allow some linguistic variability. If the percentage of OOV words in a chunk exceeds this 1.5% threshold, we conclude the current chunk and initiate a new one at the beginning of the next sentence. This approach ensures that each analyzed text chunk primarily consists of known vocabulary words, with a minimal presence of OOV words. Figure 1 illustrates our method.

After filtering, our datasets consist of around 22 billion and 2.1 billion tokens, derived from various splits of the SlimPajama dataset. The distribution of the tokens on the 22 billion dataset across various splits of the SlimPajama dataset is detailed in Table 1. For the 2.1 Billion dataset the distribution of tokens can be found in Table 5 and 6 in the

appendix.

We computed the Zipfian Coeffcient by fitting a linear regression model on the logarithm of word frequencies and ranks of words and obtained a zipfian coefficient of -1.11 indicating the dataset exhibits a distribution pattern typical of natural languages, where a small number of words are extremely common, while the majority are rare, thereby underscoring naturalistic quality of our dataset [2]. We utilize this dataset for pre-training of models we label as "simple" models (henceforce, Simple). In contrast, for our "regular" models, we train them using data from the SlimPajama dataset, applying no filtering and maintaining a similar number of tokens.

4 Experimental Setup

4.1 Model Configurations

We pre-train transformer-based models (Vaswani et al., 2017) by adapting the LLaMA architecture (Touvron et al., 2023) and vary key hyperparameters, such as the number of dimensions of the hidden representations (hidden size), number of hidden layers in the Transformer decoder (num layers) and the internal dimension of the MLP (intermediate size). We keep the base period of the RoPE embeddings (Su et al., 2023) (rope_theta) at 20.

We trained 2 models on 22 billion tokens: Simple 165M and Simple 100M. We also trained 54 models on 2.1 Billion tokens. Of these, 36 models were used for zero-shot experiments in section C of the appendix, while the remainder were utilized for curve fitting analyses in section 5.3. The hidden sizes used in our experiments are [32, 64, 128, 256, 512, 1024], with layer counts of [2, 4, 8]. For the majority of zero-shot experiments, the intermediate size was set at four times the hidden size. However, for the experiments detailed in section 5.3, we used intermediate sizes that are twice the hidden size as well. These variations in hyperparameters produced models from 1 million to 165 million parameters, as detailed in Table 7 in the appendix. For training, we utilized the Flash Attention mechanism introduced by Dao et al. (2022).

4.2 Model Pretraining

We train a tokenizer using Byte Pair Encoding (BPE) (Sennrich et al., 2015) on the filtered dataset. We use a vocabulary size of 15000. All simple models are pre-trained on a causal language modelling objective for a single epoch on the simplified data derived from various splits of SlimPajama dataset. We train two sets of models one set on 22 Billion tokens and another set on 2.1 Billion tokens. The models trained on 22 Billion tokens have an effective batch size of 512 and context lengths of 1024 with model parameters being updated 41697 times. We use a cosine learning rate scheduler with warmup and use peak learning rates in the range of 6×10^{-4} to 1×10^{-3} (learning rates are chosen based on model size) and decay the learning rate down to 13% of the peak learning rate and a perform warmup for 4000 steps. The models trained on 2.1 Billion tokens have an effective batch size of 4096 and context lengths of 128. The model parameters are updated a total of 3972 times with cosine learning rate scheduler with warmup and a peak learning rate of 2.8×10^{-3}, we decay learning rate upto 11% of the peak learning rate and perform warmup for 1000 steps.

We utilize the AdamW optimizer for updating the parameters, with the β_1, β_2, and $weight\_decay$ values of 0.9, 0.95, and 0.1 respectively. We use gradient clipping of 1.0. We conduct pre-training of all models on 2 RTX3090. For training regular models, we use a dataset consisting of 2.1 Billion tokens, and use the same hyperparameters as those used for training simple models with the same token count.

4.3 Model Evaluation and Datasets

We evaluate all pre-trained models for their zero-shot and few-shot ICL abilities using the language model evaluation harness from EleutherAI (Gao et al., 2021), which is a framework designed to perform zero-shot and few-shot evaluations. The datasets we use include: Benchmark of Linguistic Minimal Pairs for English (BLiMP) (Warstadt et al., 2020) to assess the models' capability in understanding linguistic features. BLiMP is comprised of 67 individual datasets, each containing a pair of sentences: one grammatically correct, and the other incorrect. These pairs are designed to assess the models' proficiency in recognizing morphological, syntactic, and semantic aspects of language. Additionally, we use the Physical Interaction Question Answering (PIQA) (Bisk et al., 2020) to measure the performance of lanuguage models on questions requiring physical common sense, AI2 Reasoning Challenge (ARC-Easy) (Clark et al., 2018) which

[2]This analysis was done on the 2.1B dataset

measures reasoning abilities of Language Models, Choice of Plausible Alternatives (COPA) (Roemmele et al., 2011) which evaluates common sense causal reasoning of language models, Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) which evaluates if the language model can identify if a pair of sentences constitutes a paraphrase, RTE which evaluates if language models can identify entailment and non-entailment, MultiGenre Natural Language Inference (MNLI) (Williams et al., 2017) which provides a more diverse and challenging dataset for natural language inference, and Stanford Sentiment Treebank (SST-2) (Socher et al., 2013) which evaluates the model's fine grained understanding of sentiment.

In reporting our findings, we differentiate between the entire downstream dataset called the "unfiltered dataset" or "standard dataset" and the filtered subset termed the "filtered" or "simplified dataset". The simplified version of each downstream task consists of instances using only the words from the AO-Childes lexicon (with the addition of digits and special symbols), thereby mitigating potential distributional shifts between pre-training and testing.

The primary aim of our study was to understand if smaller models could achieve performance improvements similar to those observed in larger models, but when modeling a simplified language. We would like to emphasize that this choice is motivated by our research goal: to understand whether the lack of emergent abilities in smaller pre-trained models is merely a question of model capacity, and whether reducing the problem (down-scaling the language) would allow us to see similar abilities emerge in smaller models. This logic is what directly motivates our downstream evaluation with standard datasets filtered down to imitate the constrained language setting. However, we recognize the value of evaluating model performance on unfiltered datasets for comparison. To this end, we have included the performance of simple models on unfiltered datasets as well.

5 Results

Our goal is to understand if the absence of emergent abilities in smaller pre-trained models is simply a matter of model capacity and whether simplifying the problem, i.e., downscaling the language, would allow these abilities to emerge in smaller models. To this end, we evaluate models trained on simplified data against both filtered and standard evaluation datasets. We compare our models with Pythia (Biderman et al., 2023) and OPT (Zhang et al., 2023) (models up to 1.3B parameters) to determine if downscaling the language facilitates emergent capabilities to occur much earlier.

Table 7 in the appendix shows the perplexity for different-sized models trained on both simple and regular datasets. The simple dataset is derived from a subset of the SlimPajama dataset, where the text has been filtered to limit vocabulary complexity. In contrast, the regular dataset uses the original, unaltered text from the same source. A separate test set, similar in distribution to the training data, was used to evaluate perplexity in each set of models. The reported results reflect the performance of each model trained in an identical training regimen with the same number of training steps. We find that as the model size increases, its ability to accurately predict and understand the held-out test set also improves, as evidenced by decreasing perplexity on both the simple and regular models. Furthermore, the perplexity metrics indicate that at this scale, simple models are able to learn the simple language much better. Simple models in the range of 1-165M parameters achieve perplexity of 92.00 - 20.59 on the simple dataset. In contrast, when regular models are trained on a regular dataset they achieve perplexity in the range of 193.20 - 28.97 on the regular dataset.[3]

5.1 Do simple models perform better in *zero-shot* settings?

In-context learning, as defined by Brown et al. (2020), enables models to apply knowledge gained during pre-training to new tasks without requiring fine-tuning on task-specific datasets. We evaluate the ICL capabilities of our models, focusing on their zero-shot performance across a range of tasks, including COPA, MRPC, RTE, MNLI, SST-2, PIQA, ARC-Easy, and BLiMP. These tasks are analyzed using both standard and vocabulary-filtered datasets.

Table 2 presents zero-shot performance for different models, including pre-trained Pythia models (1B, 410M, 160M), OPT models (1.3B, 350M, 125M), and models trained on simplified language (165M and 100M), which we will refer to as Simple models. The Simple models perform better on vocabulary-filtered downstream tasks than on the

[3]These results are reported on models trained on 2.1B dataset

Model	COPA	MRPC	RTE	MNLI	ARC-Easy	BLiMP	PIQA	SST-2	Average
Pythia 1B	0.72	0.68	0.53	0.34	0.57	0.84	0.71	0.50	0.61
Pythia 1B†	0.68	0.67	0.50	0.35	0.56	0.83	0.70	0.65	0.62
Pythia 410M	0.70	0.52	0.53	0.34	0.52	0.84	0.67	0.51	0.58
Pythia 410M†	0.71	0.38	0.50	0.34	0.52	0.83	0.66	0.66	0.58
Pythia 160M	0.63	0.67	0.52	0.35	0.44	0.80	0.62	0.51	0.57
Pythia 160M†	0.61	0.67	0.50	0.37	0.41	0.80	0.61	0.36	0.54
OPT 1.3B	0.79	0.66	0.51	0.36	0.57	0.86	0.72	0.82	0.66
OPT 1.3B†	0.73	0.62	0.57	0.37	0.59	0.85	0.70	0.90	0.67
OPT 350M	0.72	0.68	0.52	0.34	0.44	0.85	0.65	0.62	0.60
OPT 350M†	0.73	0.67	0.71	0.36	0.45	0.84	0.63	0.71	0.64
OPT 125M	0.66	0.68	0.50	0.34	0.44	0.83	0.63	0.53	0.58
OPT 125M†	0.61	0.67	0.50	0.34	0.47	0.83	0.63	0.42	0.56
Simple 165M	0.73	0.68	0.56	0.33	0.35	0.71	0.63	0.49	0.56
Simple 165M†	0.83	0.67	0.79	0.35	0.42	0.76	0.65	0.64	0.64
Simple 100M	0.66	0.68	0.52	0.33	0.34	0.72	0.62	0.60	0.56
Simple 100M†	0.68	0.58	0.64	0.35	0.43	0.78	0.64	0.58	0.59
Random Chance	0.50	0.50	0.50	0.33	0.25	0.50	0.50	0.50	0.45

Table 2: Zero-shot accuracy of pre-trained Pythia and OPT models vs. models trained on simplified language. Models are evaluated on both standard and vocabulary-filtered datasets. Results on vocabulary-filtered datasets are marked with †. Our findings indicate that the simplified models demonstrate superior zero-shot performance on vocabulary-filtered datasets, achieving higher average scores across these datasets compared to the average scores of significantly larger Pythia pre-trained models.

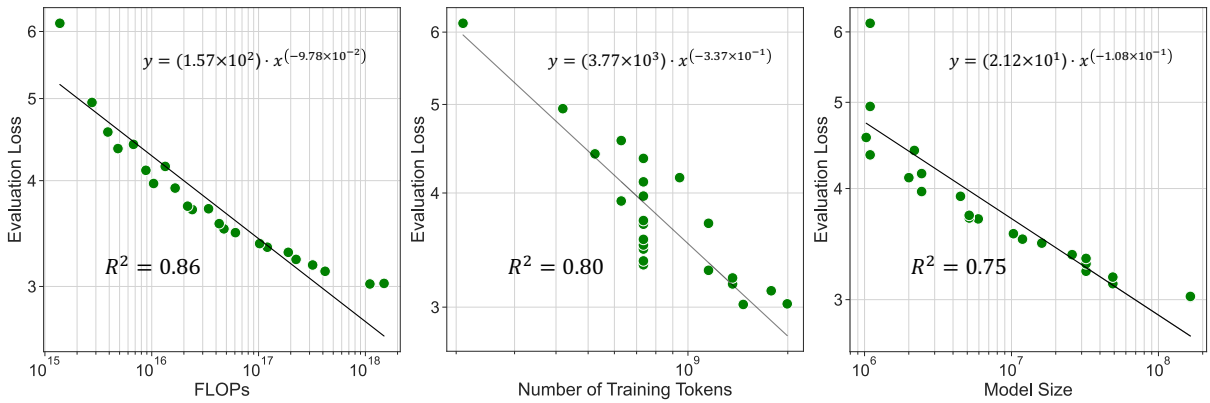


Figure 2: Here we present our curve-fitting results. The green dots represent the compute-optimal instances found in our experiments and the black solid line represents the fitted power curve of the form $y = A \cdot x^B$. In each subfigure, we provide the optimal values of A and B, and the goodness of fit (R^2). Starting from the left, we present the relationship between the evaluation loss (y-axis) and FLOPs (left subfigure), pre-training data (center subfigure) size, and model size (right subfigure), respectively. All R^2 values are over 0.74 and we observe the best fit for the left subfigure (loss vs. FLOPs). In the test range of values for model size and data size, we observe that loss value reduces faster per unit change in model size.

corresponding unrestricted standard versions of the same tasks. This is to be expected, since these models are not exposed to unrestricted language during training.

Since the larger models used in this study are pre-trained on unrestricted language, we expect them to be able to handle tasks using simplified language, as the latter is a subset of their training data. Interestingly, we see that Simple 165 model outperforms Pythia 1B model on simplified downstream data (0.64 vs. 0.62 average performance), suggesting that modeling a restricted language allows smaller models to achieve stronger-than-expected zero-shot capabilities.

A curious comparison arises between the performance of small models on simplified tasks and the performance of larger models pre-trained on standard language on the corresponding standard versions of the same tasks. In this situation, there is no distribution shift between training and testing. If both the model size and language complexity are downscaled appropriately, we expect to see similar performance figures. However, we see that the Simple 165M model performs better on simplified downstream data than the Pythia 1B on standard datasets (0.64 vs. 0.61 average performance), despite being approximately six times smaller and seeing substantially less data. We see a similar trend with OPT model family, where the Simple 165M model does better on the simplified downstream data than OPT 350M model on standard datasets (0.64 vs. 0.60 average performance).

We also report the performance of small models trained on a much smaller amount of data (2.1B tokens), comparing regular and restricted pre-training. For detailed performance comparison on the BLiMP benchmark, PIQA, and ARC-Easy datasets across different model sizes, please refer to the appendix C. As expected, pre-training smaller language models on simpler data leads to better downstream task performance. Consistent with Deshpande et al. (2023), we see above random performance of models as small as 1M parameters. Figure 4 in the appendix, shows the zero-shot task accuracy with respect to the hidden size and number of layers.

5.2 Do Simple models perform better in *few-shot* settings?

In addition to zero-shot performance, we also compare the few-shot performance of simple and standard pre-trained models. We evaluate the performance of the Simple 165M and Simple 100M models, which are pre-trained on a simplified vocabulary, against the Pythia baselines (160M, 410M, and 1B). This evaluation uses few-shot prompting using examples from vocabulary-filtered downstream data. We report the models' average performance across the following datasets: COPA, MRPC, RTE, MNLI, ARC-Easy, PIQA, and SST-2. The results for each dataset are averaged over three runs, with each run using different task examples in the context.

Model	0-shot	1-shot	2-shot	3-shot	4-shot
Pythia 1B	0.59	0.57	0.57	0.58	0.60
Pythia 410M	0.54	0.55	0.53	0.56	0.54
Pythia 160M	0.50	0.50	0.48	0.51	0.50
Simple 165M	0.62	0.56	0.56	0.54	0.54
Simple 100M	0.56	0.56	0.56	0.56	0.55

Table 3: Average few-shot results across different vocab-filtered tasks such as COPA, MRPC, RTE, MNLI, ARC-EASY, PIQA, SST. Our results reveal no discernible trend in the few-shot learning results, suggesting that larger models are required to observe the emergence of few-shot in-context learning capabilities.

From the results in Table 3, we observe no significant improvement in performance with an increased number of in-context examples. This is in line with previous findings for language models of similar sizes (Brown et al., 2020). This suggests that the smaller model sizes of 100M or 165M may not be adequate to fully demonstrate the few-shot ICL capability in the downscaled language setting. We also believe that the simplified-data models we investigated likely lacked the scale necessary to exhibit emergent abilities such as chain-of-thought prompting (Wei et al., 2022b). Just as models smaller than 10B parameters trained on unrestricted language actually perform worse with CoT prompting (Wei et al., 2022b), our simplified-data models may also require greater scale to exhibit such capabilities.

5.3 Do simple models obey power laws?

We fit a power curve of the form $L = A \cdot x^B$, to predict the cross-entropy loss (L) based on the compute cost (C), data size (D), and model size (M), separately. For curve-fitting, we consider only the 25 compute-optimal instances found for 25 bins of the FLOPs values and utilize R^2 value to assess the goodness of fit. We adopt the formula presented by Deshpande et al. (2023) to calculate the FLOPs values which considers the embedding parameters while calculating FLOPs unlike (Ka-

plan et al., 2020; Hoffmann et al., 2022). Similar to Kaplan et al. (2020); Hoffmann et al. (2022) we observe that the upstream performance (pre-training cross-entropy loss on validation split of the data) is fairly predictable with R^2 value of 0.86, 0.80, and 0.75, for compute cost, data size, and model size, respectively. We also observe that improvement in the loss value is faster for the model size compared to the data size. [4]

5.4 Do Simple Models yield good generations?

We analyze text continuations, on prompts sampled from TinyStories (Eldan and Li, 2023) and ROCStories (Mostafazadeh et al., 2016), using the 165M simple model and Pythia baselines (160M, 410M, 1B). We sample 25 different prompts from both these datasets randomly. We choose prompts from these datasets so as to keep the prompts simple enough for the model trained on vocabulary-filtered pre-training dataset to understand. For decoding of all models we set temperature to 1.0 and employ nucleus sampling (Holtzman et al., 2019) with top_p set to 0.9. The maximum number of new tokens are set to 50.

Table 4 shows few initial prompts and generations from simple models and different baselines. Similar to (Eldan and Li, 2023) we evaluate the generations with GPT-4, to assign scores ranging from 1 to 10 with 10 being the highest for different aspects of the generated text such as grammar, creativity, and coherence. We plot the average scores across all completions for each model as depicted in Figure 3. From the figure it can be seen that the simple model performs comparably to the Pythia 410M model in terms of grammar and creativity and the simple model outperforms Pythia 160M model in terms of coherence.

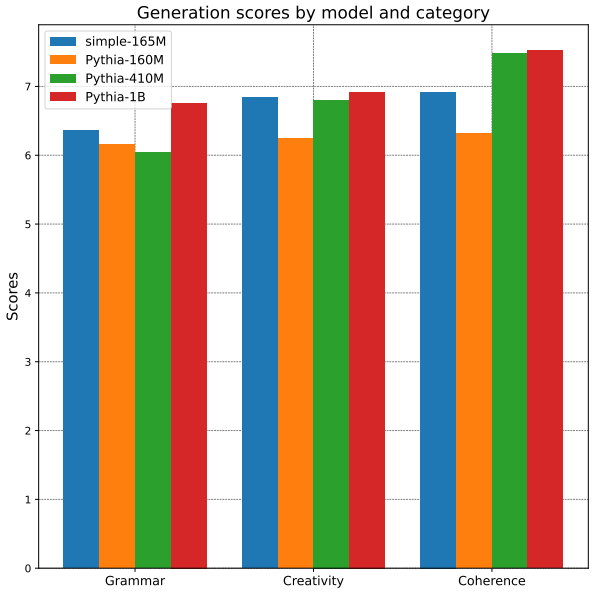


Figure 3: Comparative Analysis of Text Quality from Various Generative Models as evaluated by GPT-4. Models pre-trained on a vocabulary-simplified dataset produce outputs that are grammatical, creative, and coherent, and are comparable to those from larger models trained on the complete language dataset.

6 Conclusion

In our study, we explored the impact of simplifying pre-training data on the performance of small generative models, specifically those with fewer than 165 million parameters. Our primary focus was to assess whether these models exhibit emergent abilities, notably zero-shot learning — the capability to have non-random performance on tasks without explicit prior training. To this end, we evaluated a series of models, each varying in hidden size and the number of layers, and measured their zero-shot performance across different tasks. Our findings reveal that smaller models operating in a simplified language regime indeed demonstrate enhanced zero-shot learning abilities on vocabulary-filtered datasets, and outperform larger baseline models trained on full language on standard datasets. This suggests that the complexity of training data is a crucial factor in the development of zero-shot learning capabilities in smaller models.

We expect future work to investigate the model and data scales at which other emergent abilities (such as few-shot ICL and CoT reasoning) appear when modeling a reduced-scale language. Exploring the potential of instruction fine-tuning in models trained with simplified language is another interesting direction to pursue in future work.

7 Limitations

We adopt vocabulary reduction for simplifying the language we model and do not explore other possible ways of simplification such as sentence structure simplification, data pruning, or curriculum learning. The adopted vocabulary-based data filtration also leads to different distributions of sequence length and word frequencies compared to regular English data. Hence, our findings should be considered within the distributional properties of our pre-training data. We extend the pre-trained

[4] The results are reported on models trained on 2.1B tokens

Prompt	Simple-165M	Pythia-160M	Pythia-410M	Pythia-1B
I saw my friend Joe sitting in lobby today. I kept him company, as he is a lonely old man. He told me he had just listened to Beethoven's Ninth. I talked to him for an hour. I left him in the lobby and told him I would see him soon.	I went to check on him and he was just sitting there waiting. Then I got to see a few people who were with him that night. He had been going into his room after his accident. I went down to the lobby to see what was	He seemed to think I was the right person. We talked on the telephone for two hours. Then I came back to the room where we had been sitting at the dinner table and went out. There was a young woman sitting at the opposite end of	We went to his room. He was asleep, lying in bed, as I was. He did not wake up and I did not want him to. My friend Joe told me this afternoon that he had been on the way to visit his	He seemed to be enjoying himself. I did not leave him alone as he has been known to do. After his hour with me he decided to have a drink. He seemed very pleased with himself. I left him alone. I hope he
Todd was hungry. He did not have anything to cook at his house. He decided he need to go buy something to cook. On the way to the store Todd decided to make hamburgers. Todd buys everything he needed and goes home and cooks.	When he arrived, he saw his friends there cooking ham burgers for them. He was already thinking about dinner. He had decided he would make a ham burger. T odd looked at his friends and he knew that he had to do something about this.	When he's done cooking, he takes the ham off the rack. He makes the ham into a turkey, then adds a little salt and pepper. Todd then steaks them with chicken.The ham goes through the oven in a double skillet and	He had some leftovers left. He wanted to do this for the rest of his life.He made up for his lunch with some chips. His mother came home from work. She told him to make her hamburger. She had to eat	When he got back home he took all the food he needed. He eats all the food and it was too late to cook something else now he was back at the store. Todd decides to go in the store again. There he finds out that

Table 4: Comparison of the prompt completions generated by the Simple 165M model trained on vocabulary-filtered simplified pre-training dataset and Pythia pre-trained baselines (160M, 410M, 1B). For decoding of all models we set temperature to 1.0 and employ nucleus sampling with top_p set to 0.9. The maximum number of new tokens are set to 50.

model capabilities to process longer sequences by utilizing position interpolation method (Chen et al., 2023). We train all models with only the causal language modeling task and do not consider distillation or model pruning as a means of developing smaller models. For evaluating the pre-trained models, we focus only on the in-context learning abilities. Hence, we keep the finetuning experiments out of the scope of our study. We further note that instruction tuning of models may considerably affect the ICL abilities. However, we keep the investigation of the effect of instruction tuning on ICL abilities for future work.

Acknowledgements

We would like to thank Namrata Shivagunde for her insightful suggestions on various aspects of experiments.

References

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

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*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Vijeta Deshpande, Dan Pechi, Shree Thatte, Vladislav Lialin, and Anna Rumshisky. 2023. Honey, I shrunk the language: Language model behavior at reduced scale. *arXiv preprint arXiv:2305.17266*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2024. Understanding emergent abilities of language models from the loss perspective. *arXiv preprint arXiv:2403.15796*.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Philip A Huebner, Elior Sulem, Fisher Cynthia, and Dan Roth. 2021. Babyberta: Learning more grammar with small-scale child-directed language. In *Proceedings of the 25th conference on computational natural language learning*, pages 624–646.

Philip A Huebner and Jon A Willits. 2021. Using lexical context to discover the noun category: Younger children have it easier. In *Psychology of learning and motivation*, volume 75, pages 279–331. Elsevier.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. 2023. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, Toronto, Canada. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? *arXiv preprint arXiv:2304.15004*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Inar Timiryasov and Jean-Loup Tastet. 2023. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. *arXiv preprint arXiv:2308.02019*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2023. Opt: Open pre-trained transformer language models, 2022. *URL https://arxiv. org/abs/2205.01068*, 3:19–0.

A Data distribution for the Simple and Regular Datasets with 2B tokens

The data distribution corresponding to the simple and regular datasets containing 2B tokens can be found in 5 and 6 respectively.

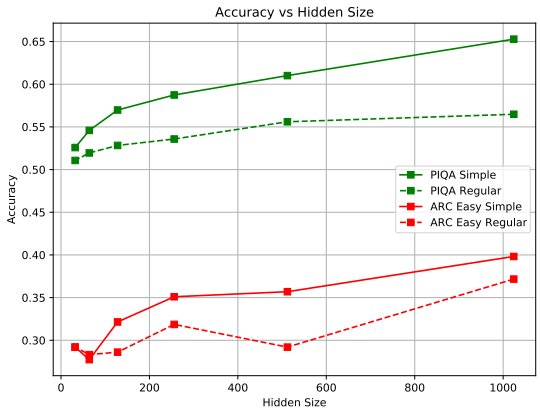
B Modeling simple and regular language

Table 7 shows the perplexity metrics across different simple and regular models trained on 2.1B tokens. Perplexity of simple models are evaluated on the held out test set of simple pre-training data whereas perplexity of regular models are evaluated on the held out test set of both simple and regular pre-training data. It can be seen that smaller models are better able to model simple data compared to regular data.

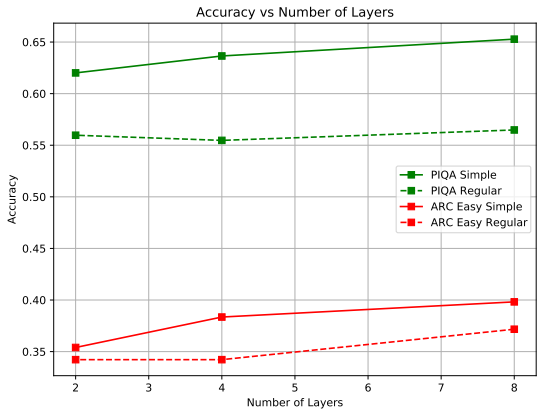
C Additional Zero-shot Evaluation Results

Figure 4 shows that the task accuracy improves with an increase in the capacity of the model for both regular and simple models. Simple models generally outperform their regular counterparts on simpler tasks such as PIQA and ARC-Easy when evaluated on the filtered dataset.

Table 8 shows additional zero-shot accuracy results across all model configurations in both filtered and unfiltered datasets. On the PIQA task, we saw that simple models typically exhibit superior performance over regular models in a majority of configurations, regardless of whether the dataset was filtered or unfiltered. When focusing on the



(a) Variation in accuracy for hidden sizes ranging from 32 to 1024 on both simple and regular models. For each hidden size, we choose the model with 8 layers.



(b) Variation in accuracy for layers ranging from 2 to 8 on both simple and regular models. For each layer we choose the model with a hidden size of 1024.

Figure 4: Model performance different hidden layer sizes and number of layers on PIQA and ARC-Easy datasets. Both simple and regular models are compared on the filtered evaluation datasets. We observe that the simple models consistently outperform their regular counterparts across different model configurations.

Split	Percentage of tokens	Number of tokens (mil)
C4	24.92%	530.877
GitHub	0.42%	885.322
Commoncrawl	18.42%	390.694
StackExchange	3.05%	649.524
Wikpedia	0.09%	186.751
ArXiv	2%	431.555
Books	51.2%	1090.048
total	100%	2130.447

Table 5: Data source distribution for the simplified pre-training dataset derived from SlimPajama.

Split	Percentage of tokens	Number of tokens (mil)
C4	26.27%	560.18
GitHub	4.86%	103.60
Commoncrawl	52.73%	1124.51
StackExchange	3.15%	67.16
Wikpedia	4.35%	92.81
ArXiv	4.62%	98.55
Books	4.02%	85.84
total	100%	2132.64

Table 6: Data source distribution for the standard SlimPajama pre-training dataset used to train regular models.

ARC-Easy filtered dataset, we found that simple models with larger hidden sizes (exceeding 64) consistently outperformed their regular counterparts on the filtered dataset. Conversely, with unfiltered ARC-Easy dataset, regular models demonstrated a higher performance level than the simple models.

Tables 9 and 10 show zero-shot accuracy results across various model configurations using the filtered and unfiltered datasets. Though simple models show a deterioration in performance compared to their results on the filtered dataset, we do see that the average scores tend to be better than the regular models on most configurations.

D Rotary Position Embeddings and Position Interpolation

For models pre-trained on 2.1B tokens we use context length of 128. However, datasets such as PIQA and ARC-Easy contain examples that span more than the pre-trained context length. To extend context window sizes beyond 128, we use Position Interpolation (Chen et al., 2023) on PIQA and ARC-Easy datasets. We use a scaling factor of 8 which allows to have context window of 1024.

Based on the findings presented in a study conducted by Liu et al. (2023), we conducted an exploratory experiment for deciding the base value for the rotary positions embeddings (Su et al., 2023). For the pre-training sequence length of 128, we observed better length extension results (with PI (Chen et al., 2023)) for the base value of 20, compared to the widely used 10,000. Our results were in agreement with the findings presented by Liu et al. (2023). Hence, we used a base value of 20 for pre-training language models.

In our downstream evaluation, we utilized PI for context length extension only for the PIQA and ARC-Easy datasets. We used a scale of 8 for extending the pre-training context length to 1024.

Model Size (M)	Hidden Size	Num Layers	Int. Size	PPL Simple (Simple Dataset)	PPL Regular (Regular Dataset)	PPL Regular (Simple Dataset)
164.96	1024	8	4096	20.59	28.97	33.85
97.84	1024	4	4096	24.59	31.13	35.49
64.28	1024	2	4096	27.92	37.15	40.34
48.92	512	8	2048	26.61	35.46	40.47
32.14	512	4	2048	28.31	38.53	42.87
23.75	512	2	2048	31.57	45.48	48.61
16.07	256	8	1024	32.72	49.18	53.58
11.87	256	4	1024	34.23	53.18	56.26
9.77	256	2	1024	37.77	62.20	62.39
5.94	128	8	512	41.38	70.05	72.53
4.89	128	4	512	43.69	77.73	77.38
4.36	128	2	512	47.37	87.18	84.05
2.44	64	8	256	54.52	102.83	100.14
2.18	64	4	256	57.86	113.23	105.72
2.05	64	2	256	62.78	124.61	114.69
1.09	32	8	128	79.61	164.78	141.03
1.02	32	4	128	84.24	178.07	150.75
0.99	32	2	128	92.00	193.20	162.80

Table 7: Parameter count and perplexity metrics across model configurations. "PPL Simple (Simple Dataset)" refers to the perplexity of simple models measured on a held-out filtered pre-training dataset. "PPL Regular (Regular Dataset)" refers to the perplexity of regular models measured on a held-out standard pre-training dataset. "PPL Regular (Simple Dataset)" refers to the perplexity of regular models measured on the held-out filtered pre-training dataset. As the model capacity increases, the ability to predict the evaluation data improves on both simple and regular models.

With the PI scale of 8, we evaluated the model on the government report dataset (Huang et al., 2021) and observed a decreasing perplexity from a context length of 64 to 1,024 in our exploratory experiment.

Hidden Size	Num. Layers	PIQA (Filtered)	PIQA (Unfiltered)	ARC Easy (Filtered)	ARC Easy (Unfiltered)
1024	8	0.653 ↑	0.642 ↑	0.398 ↑	0.330 ↓
1024	8	0.565	0.596	0.372	0.370
1024	4	0.636 ↑	0.627 ↑	0.384 ↑	0.310 ↓
1024	4	0.555	0.576	0.342	0.353
1024	2	0.620 ↑	0.621 ↑	0.354 ↑	0.309 ↓
1024	2	0.560	0.581	0.342	0.349
512	8	0.610 ↑	0.613 ↑	0.357 ↑	0.317 ↓
512	8	0.556	0.582	0.292	0.332
512	4	0.611 ↑	0.610 ↑	0.369 ↑	0.309 ↓
512	4	0.524	0.561	0.345	0.327
512	2	0.595 ↑	0.593 ↑	0.319	0.303 ↓
512	2	0.546	0.569	0.319	0.325
256	8	0.587 ↑	0.580 ↑	0.351 ↑	0.309 ↓
256	8	0.536	0.564	0.319	0.312
256	4	0.585 ↑	0.584 ↑	0.325 ↑	0.298 ↓
256	4	0.551	0.568	0.304	0.315
256	2	0.589 ↑	0.594 ↑	0.325	0.290 ↓
256	2	0.553	0.564	0.325	0.312
128	8	0.570 ↑	0.566 ↑	0.322 ↑	0.286 ↓
128	8	0.528	0.542	0.286	0.297
128	4	0.555 ↑	0.557 ↑	0.310 ↑	0.284 ↓
128	4	0.532	0.553	0.292	0.298
128	2	0.553 ↑	0.560 ↑	0.354 ↑	0.288 ↓
128	2	0.523	0.542	0.298	0.296
64	8	0.546 ↑	0.545 ↑	0.277 ↓	0.279 ↓
64	8	0.519	0.540	0.283	0.284
64	4	0.543 ↑	0.547 ↑	0.271 ↓	0.282 ↑
64	4	0.527	0.535	0.283	0.277
64	2	0.526	0.536 ↓	0.286 ↓	0.269 ↓
64	2	0.526	0.539	0.298	0.283
32	8	0.526 ↑	0.534 ↑	0.292	0.265 ↓
32	8	0.511	0.533	0.292	0.275
32	4	0.526 ↑	0.530 ↓	0.271 ↓	0.264 ↑
32	4	0.516	0.533	0.277	0.263
32	2	0.541 ↑	0.532 ↑	0.254 ↓	0.264 ↑
32	2	0.521	0.529	0.263	0.259

Table 8: Zero-shot accuracy scores on PIQA and ARC Easy dataset. "Filtered" refers to the vocabulary filtered datasets and "Unfiltered" for standard datasets. For each hidden layer size and layer count, the table compares metrics for both simple and regular models. The initial row for each hidden size and number of layers displays results from the simple model trained on simplified data, followed by a similar-sized regular model trained on regular data. Performance comparison is indicated by arrows next to the simple model's scores: a ↑ signifies the simple model outperforming the regular model, while a ↓ denotes the regular model performing better. An absence of arrows indicates comparable performance between the two models. We observe that on both the filtered and unfiltered datasets for the PIQA task, simple models generally outperform regular models in most configurations. Additionally, for the ARC Easy filtered dataset, larger simple models (with hidden sizes above 64) tend to surpass the performance of regular models. However, on the unfiltered ARC Easy dataset, it is observed that regular models outperform the simple ones.

Hidden Size	Num. Layers	Anaphor Agr.	Argument Str.	Binding	Control /Raising	Determiner Noun Agr.	Ellipsis	Filler Gap	Irregular Forms	Island Effects	NPI Lic.	Quantifiers	Subject Verb Agr.	Avg. Score
1024	8	0.968 ↑	0.804 ↑	0.711 ↓	0.792 ↑	0.932 ↑	0.806 ↑	0.741 ↑	0.871 ↑	0.555 ↓	0.662 ↑	0.847 ↑	0.843 ↑	0.794 ↑
1024	8	0.958	0.783	0.728	0.775	0.923	0.722	0.737	0.801	0.610	0.636	0.713	0.832	0.768
1024	4	0.965 ↑	0.781 ↑	0.714 ↓	0.766 ↑	0.919 ↓	0.781 ↑	0.736 ↑	0.890 ↑	0.512 ↓	0.587 ↑	0.800 ↑	0.812 ↑	0.772 ↑
1024	4	0.951	0.777	0.730	0.755	0.924	0.721	0.732	0.848	0.573	0.577	0.593	0.789	0.748
1024	2	0.961 ↑	0.793 ↑	0.697 ↑	0.778 ↑	0.918 ↑	0.736 ↑	0.729 ↑	0.923 ↑	0.488 ↓	0.616 ↓	0.805 ↑	0.783 ↑	0.769 ↑
1024	2	0.927	0.755	0.693	0.729	0.905	0.689	0.709	0.873	0.508	0.624	0.723	0.744	0.740
512	8	0.938 ↓	0.785 ↑	0.688 ↓	0.762 ↑	0.909 ↑	0.761 ↑	0.720 ↑	0.950 ↑	0.560 ↑	0.608 ↓	0.707 ↑	0.816 ↓	0.767 ↑
512	8	0.952	0.756	0.738	0.745	0.903	0.660	0.698	0.861	0.537	0.617	0.662	0.822	0.746
512	4	0.961 ↑	0.787 ↑	0.681 ↓	0.779 ↑	0.920 ↑	0.757 ↑	0.728 ↑	0.931 ↑	0.536 ↑	0.570 ↑	0.778 ↑	0.827 ↑	0.771 ↑
512	4	0.938	0.753	0.754	0.748	0.917	0.654	0.687	0.851	0.531	0.557	0.698	0.813	0.742
512	2	0.915 ↑	0.765 ↑	0.687 ↓	0.763 ↑	0.916 ↑	0.744 ↑	0.679 ↓	0.911 ↑	0.453 ↓	0.637 ↑	0.787 ↑	0.736 ↑	0.749 ↑
512	2	0.880	0.727	0.701	0.736	0.908	0.679	0.683	0.896	0.492	0.464	0.747	0.733	0.721
256	8	0.913 ↑	0.777 ↑	0.686 ↓	0.761 ↑	0.907 ↑	0.656 ↑	0.695 ↑	0.902 ↑	0.481 ↓	0.517 ↓	0.725 ↑	0.761 ↑	0.732 ↑
256	8	0.897	0.737	0.733	0.749	0.878	0.626	0.663	0.871	0.516	0.539	0.718	0.752	0.723
256	4	0.936 ↑	0.778 ↑	0.691 ↓	0.772 ↑	0.893 ↓	0.670 ↑	0.669 ↓	0.893 ↑	0.506 ↑	0.623 ↑	0.713 ↓	0.778 ↑	0.744 ↑
256	4	0.867	0.744	0.716	0.715	0.898	0.617	0.676	0.837	0.505	0.560	0.736	0.714	0.715
256	2	0.893 ↑	0.744 ↑	0.690 ↓	0.749 ↑	0.890 ↑	0.674 ↑	0.678 ↑	0.918 ↑	0.451 ↓	0.556 ↓	0.742 ↑	0.707 ↑	0.724 ↑
256	2	0.798	0.715	0.691	0.718	0.852	0.605	0.660	0.913	0.481	0.575	0.720	0.639	0.697
128	8	0.884 ↑	0.734 ↑	0.720 ↑	0.761 ↑	0.895 ↑	0.654 ↑	0.662 ↓	0.935 ↑	0.450 ↓	0.623 ↑	0.714 ↑	0.711 ↑	0.729 ↑
128	8	0.785	0.694	0.698	0.731	0.859	0.591	0.668	0.909	0.516	0.423	0.697	0.649	0.685
128	4	0.854 ↑	0.754 ↑	0.689 ↓	0.754 ↑	0.885 ↑	0.624 ↓	0.679 ↑	0.882 ↓	0.454 ↓	0.550 ↓	0.726 ↓	0.754 ↑	0.717 ↑
128	4	0.774	0.699	0.700	0.703	0.838	0.632	0.660	0.886	0.484	0.599	0.747	0.642	0.697
128	2	0.847 ↑	0.727 ↑	0.681 ↑	0.756 ↑	0.864 ↑	0.675 ↑	0.667 ↑	0.908 ↑	0.415 ↓	0.631 ↑	0.683 ↑	0.677 ↑	0.711 ↑
128	2	0.802	0.667	0.662	0.673	0.842	0.593	0.626	0.821	0.481	0.586	0.654	0.627	0.669
64	8	0.847 ↑	0.720 ↑	0.674 ↑	0.692 ↑	0.873 ↑	0.575 ↑	0.647 ↑	0.890 ↑	0.485 ↑	0.605 ↑	0.803 ↑	0.672 ↑	0.707 ↑
64	8	0.696	0.673	0.640	0.690	0.830	0.543	0.624	0.866	0.472	0.479	0.639	0.606	0.646
64	4	0.807 ↑	0.707 ↑	0.669 ↓	0.699 ↑	0.877 ↑	0.556 ↓	0.662 ↑	0.867 ↑	0.455 ↑	0.594 ↑	0.572 ↓	0.610 ↓	0.673 ↑
64	4	0.697	0.671	0.670	0.656	0.833	0.588	0.596	0.851	0.428	0.565	0.589	0.613	0.646
64	2	0.790 ↑	0.684 ↑	0.659 ↑	0.683 ↑	0.874 ↑	0.577 ↓	0.660 ↑	0.847 ↑	0.408 ↓	0.527 ↓	0.665 ↑	0.629 ↑	0.667 ↑
64	2	0.637	0.646	0.647	0.623	0.794	0.596	0.635	0.826	0.409	0.583	0.567	0.570	0.628
32	8	0.746 ↓	0.674 ↑	0.650 ↑	0.632 ↑	0.809 ↑	0.498 ↓	0.631 ↑	0.837 ↓	0.482 ↑	0.500 ↑	0.659 ↑	0.574 ↓	0.641 ↑
32	8	0.762	0.649	0.615	0.589	0.786	0.521	0.612	0.839	0.475	0.498	0.604	0.593	0.629
32	4	0.479 ↓	0.619 ↓	0.643 ↑	0.644 ↑	0.741 ↑	0.489 ↑	0.621 ↑	0.782 ↓	0.519 ↑	0.505 ↓	0.559 ↓	0.592 ↑	0.599 ↓
32	4	0.612	0.656	0.623	0.609	0.731	0.488	0.607	0.797	0.453	0.584	0.660	0.544	0.614
32	2	0.723 ↑	0.640 ↑	0.635 ↑	0.631 ↑	0.740 ↑	0.485 ↑	0.641 ↑	0.855 ↑	0.473 ↓	0.618 ↑	0.654 ↑	0.587 ↑	0.640 ↑
32	2	0.591	0.638	0.615	0.626	0.686	0.420	0.603	0.733	0.489	0.606	0.476	0.526	0.584

Table 9: Zero-shot accuracy scores on the vocabulary filtered datasets in the BLiMP benchmark. For each hidden layer size and layer count, the table compares metrics for both simple and regular models. The initial row for each hidden size and number of layers displays results from the simple model trained on simplified data, followed by a similar-sized regular model trained on regular data. Performance comparison is indicated by arrows next to the simple model's scores: a ↑ signifies the simple model outperforming the regular model, while a ↓ denotes the regular model performing better. An absence of arrows indicates comparable performance between the two models. We find that the average score of simple models tends to surpass regular models in most configurations.

Hidden Size	Num. Layers	Anaphor Agr.	Argument Str.	Binding	Control /Raising	Determiner Noun Agr.	Ellipsis	Filler Gap	Irregular Forms	Island Effects	NPI Lic.	Quantifiers	Subject Verb Agr.	Avg. Score
1024	8	0.919 ↓	0.776 ↓	0.719 ↓	0.757 ↓	0.891 ↓	0.782 ↑	0.725 ↓	0.883 ↑	0.563 ↓	0.644 ↑	0.846 ↑	0.786 ↓	0.774 ↑
1024	8	0.972	0.780	0.728	0.761	0.917	0.735	0.728	0.845	0.607	0.630	0.719	0.830	0.771
1024	4	0.918 ↓	0.749	0.709 ↓	0.742 ↑	0.870 ↓	0.753 ↑	0.726 ↑	0.885 ↑	0.502 ↓	0.568	0.797 ↑	0.762 ↓	0.748
1024	4	0.955	0.749	0.749	0.737	0.916	0.735	0.724	0.877	0.578	0.568	0.600	0.783	0.748
1024	2	0.864 ↓	0.744 ↓	0.705 ↓	0.735 ↑	0.865 ↓	0.706 ↑	0.717 ↑	0.908 ↑	0.496 ↓	0.606 ↓	0.798 ↑	0.725 ↑	0.739 ↑
1024	2	0.903	0.745	0.716	0.716	0.886	0.690	0.702	0.894	0.506	0.621	0.721	0.721	0.735
512	8	0.912 ↓	0.740 ↓	0.696 ↓	0.734 ↑	0.849 ↓	0.723 ↑	0.707 ↑	0.924 ↑	0.561 ↑	0.596 ↓	0.708 ↑	0.768 ↓	0.743 ↓
512	8	0.963	0.762	0.731	0.725	0.907	0.681	0.695	0.877	0.540	0.620	0.671	0.805	0.748
512	4	0.902 ↓	0.735 ↓	0.677 ↓	0.744 ↑	0.869 ↓	0.710 ↑	0.720 ↑	0.912 ↑	0.537 ↑	0.566 ↑	0.755 ↑	0.769 ↓	0.741 ↑
512	4	0.942	0.757	0.733	0.743	0.885	0.652	0.684	0.892	0.528	0.551	0.704	0.800	0.739
512	2	0.855 ↓	0.714 ↓	0.690 ↓	0.727 ↑	0.862 ↓	0.685 ↑	0.674 ↓	0.887 ↑	0.443 ↓	0.628 ↑	0.787 ↑	0.674 ↓	0.719 ↑
512	2	0.856	0.725	0.715	0.724	0.881	0.681	0.676	0.883	0.488	0.473	0.757	0.709	0.714
256	8	0.856 ↓	0.725 ↓	0.697 ↓	0.728	0.863 ↑	0.625 ↑	0.690 ↑	0.874 ↑	0.495 ↓	0.509 ↓	0.729 ↑	0.704 ↓	0.708 ↓
256	8	0.899	0.733	0.751	0.728	0.856	0.621	0.670	0.873	0.518	0.536	0.722	0.747	0.721
256	4	0.815 ↓	0.724 ↓	0.713 ↓	0.731 ↑	0.838 ↓	0.650 ↑	0.681 ↑	0.891 ↑	0.509 ↓	0.616 ↑	0.705 ↓	0.712 ↑	0.715 ↑
256	4	0.870	0.729	0.728	0.700	0.864	0.614	0.679	0.853	0.515	0.547	0.740	0.695	0.711
256	2	0.818 ↑	0.689 ↓	0.706 ↓	0.711 ↑	0.821 ↑	0.636 ↑	0.675 ↑	0.892 ↓	0.441 ↓	0.552 ↓	0.737 ↑	0.656 ↑	0.695 ↑
256	2	0.743	0.700	0.707	0.696	0.812	0.627	0.665	0.894	0.486	0.559	0.732	0.627	0.687
128	8	0.772 ↑	0.674 ↓	0.728 ↑	0.718 ↑	0.823 ↓	0.593 ↑	0.665 ↓	0.884 ↓	0.452 ↓	0.606 ↑	0.727 ↑	0.662 ↑	0.692 ↑
128	8	0.760	0.696	0.717	0.702	0.829	0.584	0.669	0.893	0.517	0.422	0.709	0.634	0.678
128	4	0.751 ↑	0.685 ↓	0.698 ↓	0.706 ↑	0.830 ↑	0.606 ↑	0.678 ↑	0.859 ↓	0.458 ↓	0.544 ↓	0.720 ↓	0.692 ↑	0.686 ↑
128	4	0.695	0.69	0.727	0.686	0.807	0.594	0.664	0.881	0.484	0.588	0.751	0.622	0.682
128	2	0.752 ↑	0.659 ↓	0.696 ↑	0.707 ↑	0.795 ↑	0.608 ↑	0.660 ↑	0.875 ↑	0.432 ↓	0.617 ↑	0.696 ↑	0.609 ↑	0.676 ↑
128	2	0.738	0.668	0.676	0.663	0.786	0.562	0.645	0.833	0.476	0.582	0.667	0.594	0.658
64	8	0.692 ↑	0.650 ↓	0.699 ↑	0.675 ↑	0.808 ↑	0.556 ↑	0.663 ↑	0.867 ↑	0.493 ↑	0.593 ↑	0.794 ↑	0.607 ↑	0.675 ↑
64	8	0.597	0.670	0.664	0.658	0.781	0.528	0.642	0.859	0.465	0.470	0.645	0.586	0.630
64	4	0.638 ↑	0.642 ↓	0.680 ↓	0.686 ↑	0.807 ↑	0.506 ↓	0.665 ↑	0.845 ↑	0.446 ↑	0.578 ↑	0.575 ↓	0.567 ↓	0.636 ↑
64	4	0.569	0.657	0.692	0.651	0.783	0.541	0.620	0.829	0.420	0.557	0.598	0.581	0.625
64	2	0.634 ↑	0.626 ↓	0.678 ↑	0.668 ↑	0.809 ↑	0.536 ↓	0.661 ↑	0.831 ↑	0.408 ↑	0.519 ↓	0.664 ↑	0.591 ↑	0.635 ↑
64	2	0.585	0.637	0.659	0.620	0.752	0.590	0.638	0.816	0.400	0.585	0.565	0.551	0.617
32	8	0.740 ↑	0.615 ↓	0.672 ↑	0.619 ↑	0.744 ↑	0.461 ↓	0.635 ↑	0.818 ↓	0.465 ↓	0.499 ↑	0.660 ↑	0.536 ↓	0.622 ↑
32	8	0.513	0.630	0.641	0.592	0.709	0.498	0.611	0.831	0.469	0.496	0.602	0.557	0.596
32	4	0.392 ↓	0.575 ↓	0.655 ↑	0.613 ↑	0.707 ↑	0.461 ↓	0.625 ↑	0.767 ↓	0.494 ↑	0.500 ↓	0.565 ↓	0.543 ↑	0.575 ↓
32	4	0.539	0.635	0.653	0.606	0.663	0.477	0.618	0.800	0.451	0.577	0.659	0.535	0.601
32	2	0.567 ↑	0.587 ↓	0.663 ↑	0.614 ↑	0.688 ↑	0.432 ↑	0.646 ↑	0.831 ↑	0.468 ↓	0.621 ↑	0.647 ↑	0.533 ↑	0.608 ↑
32	2	0.499	0.616	0.633	0.599	0.625	0.424	0.609	0.759	0.486	0.598	0.480	0.513	0.570

Table 10: Zero-shot accuracy scores on the datasets in the BLiMP benchmark. For each hidden layer size and layer count, the table compares metrics for both simple and regular models. The initial row for each hidden size and number of layers displays results from the simple model trained on simplified data, followed by a similar-sized regular model trained on regular data. Performance comparison is indicated by arrows next to the simple model's scores: a ↑ signifies the simple model outperforming the regular model, while a ↓ denotes the regular model performing better. An absence of arrows indicates comparable performance between the two models. We find that the average score of simple models tends to surpass regular models in most configurations.