Demystifying Mixed Outcomes of Self-Training: Pre-training Analyses on Non-Toy LLMs

Anonymous ACL submission

Abstract

We investigate whether large language models (LLMs) can improve through recursive training on self-generated text, a topic where prior studies report conflicting outcomes: some find evidence of performance gains (i.e., *self-improvement*), while others observe performance degradation (i.e., *model collapse*). To clarify this discrepancy, we use the OLMo-2 models as non-toy LLMs and perform multiple rounds of continual pre-training using self-generated text with different prompting strategies and data filtering. Our experiments show that naive recursive self-training does not improve either perplexity or downstream task performance, regardless of model size. These results suggest that model collapse observed in naive recursive training is inherent to the training procedure itself, while self-improvement likely owes its success not to the model's autonomous refinement but to human-designed, strategic synthetic pipelines that inject external intelligence.

1 Introduction

Training large language models (LLMs) on LLM-generated synthetic data has become an increasingly popular research paradigm (Tan et al., 2024; Maini et al., 2024; DatologyAI et al., 2025). A typical approach uses a sufficiently powerful model to produce synthetic text and then train a smaller, more efficient model on this data. However, this raises a fundamental question: who generates the synthetic data for improving the powerful model itself? This question naturally leads to the idea of recursive training, in which a model is continually trained on its own generated text.

Recent studies on recursive training report seemingly contradictory outcomes (Shumailov et al., 2024; Huang et al., 2022; Li et al., 2024; Shang et al., 2025; Wang et al., 2025a; Drayson et al., 2025; Herel and Mikolov, 2024; Wang et al., 2025b). Some claim self-improvement, where the model's performance continues to increase through iterative recursive training. Others warn of model collapse, where the model performance deteriorates over iterations. Understanding the cause of this discrepancy is the focus of this work.

A closer look at prior research reveals that these opposing findings often stem from different experimental designs. Studies reporting self-improvement typically employ strong, non-toy LLMs, adopt strategic data generation pipelines (e.g., crafted prompts, task-aware sampling, and/or quality filtering), and evaluate improvements on downstream tasks. In contrast, studies reporting model collapse usually rely on small toy models, use naive generation procedures (e.g., unconditional text generation), and evaluate degradation in perplexity on the original pre-training corpus. Such variations obscure whether the success or failure of recursive training arises from the initial model quality, the data synthesis pipeline, or the evaluation metrics.

To disentangle these factors, we conduct a systematic investigation using non-toy LLMs while deliberately adopting a simple data synthesis pipeline. Specifically, we perform multiple rounds of recursive, continual pre-training on the OLMo-2 family (OLMo et al., 2025). We examine the effects of data synthesis strategy (from empty vs. few-token prompts) and quality filtering (based on perplexity thresholds) on both downstream task performance and perplexity on the original training corpus.

Experiments show that, across all conditions, the model performance consistently degrades in both metrics, indicating model collapse rather than self-improvement. These results suggest that model collapse observed in naive recursive training is not merely due to weak base models or evaluation metric choices, but rather inherent to the training procedure itself. Conversely, studies that report self-improvement likely owe their success not to the model's autonomous refinement, but to human-

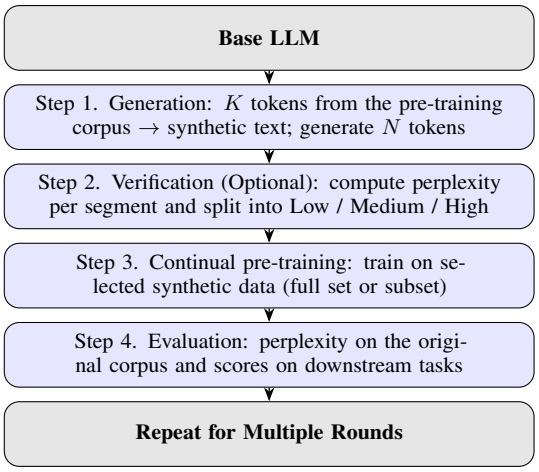


Figure 1: Overview of the continual pre-training using self-generated text pipeline

designed, strategic synthetic pipelines that inject external intelligence into the loop.

2 Problem Setting

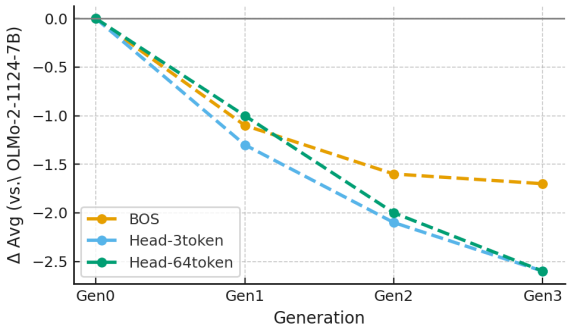
We study recursive pre-training, in which an LLM generates text that is then used to further pre-train itself. Figure 1 illustrates the pipeline. We start with a capable, non-toy base LLM. In each round, the current model repeatedly samples a prefix of $K \in \mathbb{Z}_{\geq 0}$ tokens[1] uniformly at random from the pre-training corpus and generates the continuation, until a total of $N \in \mathbb{N}$ synthetic tokens is accumulated. To examine the effect of quality consideration, we optionally apply perplexity-based filtering, which partitions generated texts into low/middle/high-perplexity tiers. When this filtering is enabled, the model over-generates $3N$ tokens, and we retain N tokens according to perplexity. The resulting synthetic corpus is then used for continual pre-training, updating the model parameters.

At the end of each round, we evaluate the current model's perplexity on the original pre-training corpus and its performance on downstream tasks.

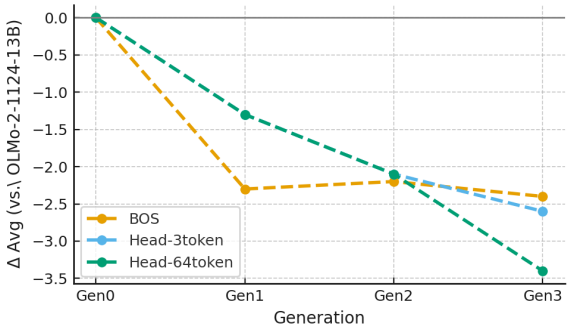
3 Experiments

We conducted experiments using the OLMo-2 family (OLMo et al., 2025), a suite of non-toy LLMs that deliver competitive performance on diverse downstream tasks. Crucially, OLMo-2 provided access to its pre-training corpora, allowing us to assess the perplexity on the original training data.

[1] $K = 0$ indicates an empty prompt.



(a) Relative performance against the initial 7B checkpoint.



(b) Relative performance against the initial 13B checkpoint.

Figure 2: Average downstream task score shift across generations.

This accessibility was an essential requirement for investigating our research question.

3.1 Training Setup

We used the OLMo-2 family (OLMo et al., 2025) for our experiments. Specifically, we employed the 7B-parameter[2] and 13B-parameter[3] models. These models were trained through a two-stage pre-training process; in Stage-1, the model was pre-trained from scratch on the OLMo-Mix-1124 corpus[4]; in Stage-2, continual pre-training (also known as mid-training) was conducted on the Dolmino-Mix-1124 corpus[5]. Unless otherwise noted, we used the model checkpoints obtained after the Stage-2 pre-training. For each round, we generated one billion tokens (i.e., $N = 10^9$) in total to train the next generation model. To generate each instance, we conditioned the model on a prompt of $K = \{0, 3, 64\}$ tokens sampled from the pre-training corpus (with $K = 0$ indicating an

[2] https://huggingface.co/allenai/OLMo-2-1124-7B

[3] https://huggingface.co/allenai/OLMo-2-1124-13B

[4] https://huggingface.co/datasets/allenai/olmo-mix-1124

[5] https://huggingface.co/datasets/allenai/dolmino-mix-1124

Model / Setting	MMLU	ARC C	HSwag	WinoG	NQ	DROP	**Avg**	PPL	
								OLMo	DOLMino
OLMo-2-1124-7B									
Base (no continual pre-training)	63.7	79.8	83.8	77.2	36.9	60.9	67.1	11.038	8.415
BOS (Gen1)	63.4	79.4	82.0	76.8	35.5	58.6	66.0	11.340	8.580
BOS (Gen2)	63.0	79.1	82.3	76.2	34.2	58.1	65.5	11.654	8.726
BOS (Gen3)	62.9	78.2	82.0	76.6	33.4	59.0	65.4	11.836	8.813
Head-3token (Gen1)	63.0	80.0	84.5	77.0	32.8	57.6	65.8	11.643	8.866
Head-3token (Gen2)	63.0	79.8	84.4	75.1	31.8	56.1	65.0	11.790	8.943
Head-3token (Gen3)	62.3	79.2	83.8	75.2	30.5	56.0	64.5	11.987	9.079
Head-64token (Gen1)	63.5	80.0	83.0	76.3	33.5	60.1	66.1	11.498	8.653
Head-64token (Gen2)	63.1	78.8	82.3	75.6	32.7	57.9	65.1	11.791	8.831
Head-64token (Gen3)	62.3	79.2	83.8	75.2	30.5	56.0	64.5	12.011	9.025
Verification split: Low-PPL	63.2	79.1	82.6	76.1	35.9	59.9	66.1	11.432	8.578
Verification split: Mid-PPL	63.2	78.8	82.5	75.5	34.6	59.5	65.7	11.374	8.593
Verification split: High-PPL	63.3	79.4	83.1	76.1	34.8	58.2	65.8	11.398	8.648
OLMo-2-1124-13B									
Base (no continual pre-training)	67.5	83.5	86.4	81.5	46.7	70.7	72.7	9.607	7.042
BOS (Gen1)	65.8	84.0	86.9	78.8	38.9	68.1	70.4	9.316	6.746
BOS (Gen2)	65.8	83.4	86.7	79.2	40.5	67.3	70.5	9.955	7.564
BOS (Gen3)	65.6	83.2	86.2	78.8	40.1	67.7	70.3	10.242	7.894
Head-3token (Gen1)	66.3	83.2	85.2	79.8	43.2	70.8	71.4	9.635	7.114
Head-3token (Gen2)	66.0	82.8	84.1	79.6	40.6	70.5	70.6	9.861	7.360
Head-3token (Gen3)	65.4	82.2	83.4	79.7	39.8	69.9	70.1	10.072	7.565
Head-64token (Gen1)	66.5	83.2	85.3	79.7	42.3	71.4	71.4	9.633	7.119
Head-64token (Gen2)	66.3	82.7	84.0	80.0	40.6	69.8	70.6	9.883	7.360
Head-64token (Gen3)	65.7	81.7	83.0	78.8	38.6	68.0	69.3	10.116	7.590

Table 1: Downstream performance and perplexity on the pre-training corpora. Rows are grouped by the initial checkpoint. In each group, the first row reports the initial model; the indented rows are results after applying continual pre-training to that same checkpoint by different generation/verification settings and rounds.

Model / Setting	MMLU	ARC C	HSwag	WinoG	NQ	DROP	**Avg**	PPL	
								OLMo	DOLMino
OLMo-2-1124-7B (Stage-1)									
Stage-1 (no continual pre-training)	59.8	72.6	81.3	75.8	29.0	40.7	59.9	11.434	8.900
BOS (Gen1) from Stage-1	59.9	72.0	81.5	75.1	29.0	40.9	59.7	11.752	9.066
Real-corpus (dolmino-mix, Gen1)	60.9	73.8	82.1	76.2	29.3	41.0	60.6	11.104	8.822
OLMo-2-1124-13B (Stage-1)									
Stage-1 (no continual pre-training)	63.4	80.2	84.8	79.4	34.5	49.6	65.3	10.500	7.913
Real-corpus (dolmino-mix, Gen1)	66.7	84.0	85.6	78.9	41.5	70.2	71.2	10.060	7.787

Table 2: Downstream performance and corpus perplexity, using the Stage-1 checkpoints for initialization.

empty prompt). We performed recursive continual training up to three rounds for each setting.

We followed the training configurations employed in the Stage-2 pre-training of the OLMo-2 family. Specifically, we employed the AdamW optimizer with peak learning rates of 6e-5 for the 7B models and 9e-5 for the 13B models, scheduled to decay to zero by the end of training.

We ran experiments on a GPU cluster in which each nodes had eight NVIDIA H200 GPUs. Generating one billion tokens took less than one node-day for both 7B and 13B models. Continual pre-training on 1B tokens required again less than one node-day for both 7B and 13B models.

3.2 Evaluation Criteria

We evaluate the performance on downstream tasks as well as perplexity on the original training corpus. As for the downstream tasks, we targeted the following benchmark datasets: MMLU (Hendrycks et al., 2020), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), Natural Questions (Kwiatkowski et al., 2019), and Drop (Dua et al., 2019). We evaluated the perplexity on the OLMo-Mix-1124 corpus and the Dolmino-Mix-1124 corpus.

3.3 Results

Table 1 summarizes our main results, while Figure 2 illustrates the score shift across generations for each setting. We observed that recursive self-training failed to improve model performance in all cases, indicating model collapse rather than self-improvement. Regardless of the choice of K token prefixes, downstream task accuracy did not increase, and perplexity either remained unchanged or worsened. Additionally, filtering or sorting generated data by perplexity did not contribute to performance gains.

4 Analyses

4.1 Control Experiments

We conducted control experiments to disentangle whether observed performance changes stemmed from the nature of generated data or other factors (e.g., an overly high learning rate). Concretely, starting from the Stage-1 checkpoints, we continued pre-training on 1B tokens sampled from Dolmino-Mix-1124, the corpus used for Stage-2 pre-training. We contrasted this with continued pre-training on a 1B-token synthetic corpus generated from the Stage-1 checkpoints using our pipeline with $K = 0$. Apart from the corpus, all hyper-parameters and procedures were the same as those described in Section 3.1.

Table 2 shows the results. For the Stage-1 models, training on synthetic data again decreased accuracy, whereas continual pre-training on the actual corpus improved performance across all downstream tasks and reduced perplexity. These results indicate that the failure of self-improvement arises from the quality of the synthetic data rather than the training configurations.

4.2 Analysis on Diversity

Motivated by the observation that perplexity-based sorting did not translate into accuracy gains, we analyzed the generated text from a different angle by measuring its diversity. To quantify the diversity of generated corpora, we used the Multilingual-E5 sentence embedding model (Wang et al., 2024). For each synthetic corpus, we encoded its texts into dense vectors with the the embedding model and computed the cosine similarity matrix between all embeddings. Let S be the set of off-diagonal, upper-triangular cosine similarities. Our diversity score is:

$$\text{Diversity} = 1 - \text{mean}(S),$$

Model	mean	median	min	max
OLMo-2-1124-7B	0.2423	0.2333	0.0000	0.5053
BOS (Gen1)	0.2340	0.2268	0.0000	0.4873
BOS (Gen2)	0.2258	0.2195	0.0000	0.5021
BOS (Gen3)	0.2182	0.2132	0.0000	0.4774
Head-3token (Gen1)	0.2221	0.2191	0.0000	0.4697
Head-3token (Gen2)	0.2105	0.2081	0.0000	0.4503
Head-3token (Gen3)	0.2001	0.1980	0.0000	0.4297
Head-64token (Gen1)	0.2246	0.2218	0.0000	0.4692
Head-64token (Gen2)	0.2120	0.2094	0.0000	0.4630
Head-64token (Gen3)	0.1947	0.1927	0.0000	0.4241
OLMo-2-1124-7B (low-ppl)	0.2456	0.2383	0.0000	0.4837
OLMo-2-1124-7B (mid-ppl)	0.2412	0.2318	0.0000	0.4773
OLMo-2-1124-7B (high-ppl)	0.2141	0.2097	0.0028	0.4329
OLMo-2-1124-13B	0.2365	0.2313	0.0000	0.4838
BOS (Gen1)	0.2475	0.2392	0.0000	0.4911
BOS (Gen2)	0.2313	0.2268	0.0000	0.4854
BOS (Gen3)	0.2312	0.2268	0.0000	0.4854
Head-3token (Gen1)	0.2248	0.2219	0.0000	0.4624
Head-3token (Gen2)	0.2113	0.2087	0.0000	0.4532
Head-3token (Gen3)	0.2012	0.1989	0.0000	0.4334
Head-64token (Gen1)	0.2246	0.2218	0.0000	0.4692
Head-64token (Gen2)	0.2120	0.2094	0.0000	0.4630
Head-64token (Gen3)	0.2016	0.1993	0.0000	0.4469

Table 3: Diversity of generated texts

the complement of the average pairwise similarity (higher is more diverse). In addition, we report the statistics (mean, median, min, max) for descriptive analysis. Table 3 shows the results. While a few settings showed increases in the diversity score, most generation methods exhibited a decline in diversity. This pattern suggests that the quality of the generated text may be deteriorating in many cases. Moreover, providing a portion of the pre-training corpus as a prompt did not consistently raise diversity relative to using an empty prompt, indicating that partial context injection may be insufficient to reliably maintain or enhance the diversity of resultant synthetic corpora.

5 Conclusion

In this work, we investigated the potential for self-improvement in non-toy large language models through simple recursive pre-training on self-generated data, conducting experiments across multiple model sizes and prompting strategies. Our results consistently showed that self-improvement did not occur under any of the tested methods. These results suggest that model collapse observed in naive recursive training is inherent to the training procedure itself, wheres self-improvement likely owes its success not to the model's autonomous refinement but to human-designed, strategic synthetic pipelines that inject external intelligence.

6 Limitations

Our experiments were conducted using open-source models from the OLMo-2 family, selected for their high reproducibility and accessibility. While these models are among the strongest open models currently available, they remain substantially smaller than state-of-the-art closed models such as GPT-5 or Claude 4. Consequently, our findings may not fully capture the behaviors of larger proprietary systems, which could exhibit different dynamics in self-improvement or model collapse.

References

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 preprint arXiv:1803.05457*.

DatologyAI, :, Pratyush Maini, Vineeth Dorna, Parth Doshi, Aldo Carranza, Fan Pan, Jack Urbanek, Paul Burstein, Alex Fang, Alvin Deng, Amro Abbas, Brett Larsen, Cody Blakeney, Charvi Bannur, Christina Baek, Darren Teh, David Schwab, Haakon Mongstad, and 12 others. 2025. Beyondweb: Lessons from scaling synthetic data for trillion-scale pretraining. *Preprint*, arXiv:2508.10975.

George Drayson, Emine Yilmaz, and Vasileios Lampos. 2025. Machine-generated text detection prevents language model collapse. *Preprint*, arXiv:2502.15654.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

David Herel and Tomas Mikolov. 2024. Collapse of self-trained language models. *Preprint*, arXiv:2404.02305.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *Preprint*, arXiv:2210.11610.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Siheng Li, Cheng Yang, Zesen Cheng, Lemao Liu, Mo Yu, Yujiu Yang, and Wai Lam. 2024. Large language models can self-improve in long-context reasoning. *Preprint*, arXiv:2411.08147.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. 2024. Rephrasing the web: A recipe for compute and data-efficient language modeling. *Preprint*, arXiv:2401.16380.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2025. 2 olmo 2 furious. *Preprint*, arXiv:2501.00656.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Ziyu Shang, Jianghan Liu, Zhizhao Luo, Peng Wang, Wenjun Ke, Jiajun Liu, Zijie Xu, and Guozheng Li. 2025. Acquisition and application of novel knowledge in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18348–18368, Vienna, Austria. Association for Computational Linguistics.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, Miami, Florida, USA. Association for Computational Linguistics.

Lecheng Wang, Xianjie Shi, Ge Li, Jia Li, Xuanming Zhang, Yihong Dong, Wenpin Jiao, and Hong Mei. 2025a. Theoretical proof that auto-regressive language models collapse when real-world data is a finite set. *Preprint*, arXiv:2412.14872.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Ze Wang, Zekun Wu, Jeremy Zhang, Xin Guan, Navya Jain, Skylar Lu, Saloni Gupta, and Adriano Koshiyama. 2025b. Bias amplification: Large language models as increasingly biased media. *Preprint*, arXiv:2410.15234.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.