Can Instruct-Tuned Models Learn New Things?

Scott Susanto scottsus@usc.edu
Clement Chan cjchan@usc.edu
Trevor Asbery asbery@usc.edu
Bruno Segovia segovia@usc.edu

Abstract

The meteoric adoption rate of Large Language Models (LMs) is widely attributed to their ability to recall important facts, encountered during training, and present them well in the manner of a human assistant or chatbot. An LM first learns facts during its pre-training phase and then learns the behavior of an assistant during its instruct-tuning phase, necessarily in that order. After the instruct-tuning phase, further injection of knowledge similar to the pre-training style will corrode its assistant-like behavior, necessitating further instruct-tuning. In this work we explore a novel technique inspired by human ways of learning new facts, utilizing both raw information and flashcard-style questions, attempting to teach instruct-tuned models new information without losing their conversational behavior. We evaluate 2 types of models, Mamba-2.8b and Gemma-2.5b, across 3 types of commonly desired LM tasks: learning a codebase, understanding research papers, and memorizing a bank of products and descriptions. We also evaluate the performance of fine-tuning against the industry standard RAG method, which provides good recall capabilities but with a more complex architecture involving a vector store retriever. From our experiments, we observe that Mamba-2.8b can in fact learn new factual knowledge while still retaining its conversational behavior, confirming our initial hypothesis that instruct-tuned models can indeed continue to learn. [1]

1 Introduction

A modern LM experiences a 2–3 step process to go from a model with randomly initialized weights to a full conversational assistant.

Pre-training The first phase is pre-training, where the model is trained with a large corpus of text, with a size usually on the order of trillions of tokens. This forms a base model whose output is reminiscent of an endless auto-complete functionality. Interestingly, by drawing on the relationships between training tokens, an emergent property is observed: the LM learns facts about the training data, and this is where the true *learning* of machine learning takes place.

Continual Pre-training Despite the LM's impressive ability to learn massive amounts of data, there are a few limitations to this knowledge, most relevant to our work being that the knowledge learned is static and does not update with time. To alleviate this problem, prior research has shown that it is possible to update a model's knowledge with new information without repeating the entire pre-training process from scratch [5]. In a process they coinced *continual pre-training*, Gupta et al. used an existing model checkpoint and continued the unsupervised fine-tuning process similar to the model's initial pre-training phase.

Instruct-tuning After pre-training, instruction-tuning (instruct-tuning for brevity) is performed. In this phase, the model is trained with text resembling a conversation, usually with question–answer pairs or instruction–question–answer tuples. Here the learning rate is adjusted to be lower than that

[1] Source code for dataset construction and fine-tuning can be found at `https://github.com/scottsus/perpetual`

Preprint. Under review.

during pre-training to preserve the generalization capabilities learned during pre-training. This is where a base model learns to answer questions in a conversational manner, optimal for user-facing chatbots. Even though instruct-tuning is orders of magnitude less computationally intensive than pre-training, it remains a relatively time-consuming and expensive process, and it is generally extremely undesirable to repeat this process.

The Instruct-tuned Angle of Continual Learning Given that LMs only learn things during pre-training but need to be instruct-tuned afterwards, a natural question arises: *Can instruct-tuned models experience continual learning as pre-trained models do?*

2 Background

In the field of machine learning, learning might mean a number of different things, and it is imperative we define learning in our context.

Learning in the human sense We first eliminate ambiguities related to learning in the ML sense, such as updating learnable parameters, adjusting learning rates, etc., and strictly confine learning to the human sense. This is still a complex philosophical task, but if we further restrict learning to the memorization and recall of factual knowledge encountered during training, we can create an entire knowledge base of singular facts with which to test the LM.

Note that it is imperative that the knowledge base contains factual knowledge that the model has not already seen during its prior training - otherwise, a seemingly capable model may only be recalling instead of learning.

Definitions Mathematically, let $\mathcal{Q} = \{q_n\}_{n=1}^{N}$ be a sequence of N multiple choice factual questions derived from the knowledge base, each having 4 options and exactly 1 correct answer. For each question q_n, let $\mathcal{O}_n = \{o_n^1, o_n^2, o_n^3, o_n^4\}$ be a set of possible options and c_n the corresponding correct answer.

Let $\mathcal{M}$ be a regular instruct-tuned model. We denote $a_n = \mathcal{M}(q_n) \in \{a_n^1, a_n^2, a_n^3, a_n^4\}$ as the predicted answer of the model at the n^{th} question.

We can then define a model $\mathcal{M}$'s accuracy score $\mathcal{L}$ on questions $\mathcal{Q}$ as

$$\mathcal{L}_{\mathcal{M},\mathcal{Q}} := \frac{\text{count}(q_n | a_n = c_n)}{N}$$

Building on this, let $\mathcal{M}'$ be the instruct-tuned model that has undergone additional learning on the train split of $\mathcal{Q}$. We determine that the model successfully learned new factual knowledge if the following proposition holds:

$$\mathcal{L}_{\mathcal{M}',\mathcal{Q}} > \mathcal{L}_{\mathcal{M},\mathcal{Q}}$$

In simpler terms, we say that the instruct-tuned LM $\mathcal{M}'$ has learned new information in the human sense if after undergoing further training, it gets more correct answers than its untrained version $\mathcal{M}$ on the same set of questions $\mathcal{Q}$, i.e. same knowledge base.

3 Related Work

Continual Pre-training: Gupta et al. explored the performance of LMs undergoing continual pre-training through unsupervised fine-tuning. Starting out with an original knowledge base $\mathcal{O}$ and an auxiliary knowledge base $\mathcal{E}$, they set a lower bound for performance with an untrained model trained only with $\mathcal{E}$ and an upper bound with an untrained model trained with $\mathcal{O} \cup \mathcal{E}$. As expected, the model retained most of its prior knowledge of $\mathcal{O}$ while learning new knowledge of $\mathcal{E}$ [5].

Instruction Tuning: Contrary to unsupervised fine-tuning, Lee et al. explored the possibility of supervised fine-tuning for not just learning facts but understanding difficult concepts. Leaning on Bloom's taxonomy, the authors used an *expert* to generate question–answer pairs of increasing cognitive difficulty on various subjects like mathematics, hard sciences, humanities, etc. Their teaching method, CORGI, achieved substantial improvements in performance, achieving gains on multiple benchmarks like the MMLU, OpenbookQA, Arc-hard, and several others [4].

Mamba: Given the emphasis on fine-tuning and the need for fast inference and training times in this paper, we explore alternatives to the traditional Transformer architecture, which has a quadratic time complexity in sequence length. In the face of recent numerous attempts to alleviate the problems related to the attention mechanism, a standout solution is the notion of Structured State Space Models (SSMs). Not only does an SSM utilize a selection mechanism that allows it to filter out input data based on its contextual relevance, it also uses a hardware-aware algorithm for efficiency. An SSM like Mamba is about 5 times faster to train than transformers of similar size, which for our purposes accelerates the fine-tuning process [1].

Retrieval Augmented Generation (RAG): The two primary methods of injecting factual knowledge in an LM, fine-tuning and retrieval, have been extensively compared [7], with RAG consistently outperforming unsupervised fine-tuning for memorization and recall. Ovadia et al. use the following analogy: *Who would do better? A student who studied the night before a closed-book exam, or a student taking an open-book exam?* As a result, there has been a massive increase in various industry standard closed– and open–source RAG solutions offered for enterprises of all scales, most of them for the purpose of memorizing and recalling information from an external knowledge base, including data sources like company catalogues and other non-sensitive, public information. Unfortunately, RAG comes with its own set of drawbacks [9]; in particular, the addition of the retriever introduces additional mechanical complexity to the system.

4 Methodology

From a practical standpoint, we consider the potential use cases of fine-tuning LMs with new knowledge, producing three standout ways current end-users could utilize this feature: a code copilot, a continually updating research assistant, and an enterprise customer support chatbot providing product information. First, for the code copilot, we used a random open-source project unseen by any model during their training. Second, for the research assistant, we trained it on 10 papers published after November 2023, beyond the cutoff date for the information used to train the models. And third, for the enterprise chatbot, we used a set of product descriptions from the WDC products dataset.

4.1 Intuition

Flash card questions To preserve the model's assistant-like behavior while learning crucial facts, we propose a novel idea in this paper: a mix of unsupervised and supervised fine-tuning in the same training cycle. Intuitively, just as humans learn by exposure to raw text then self-reflection in the form of flash cards and exams, we explore the possibility of this behavior transferring to LMs as well.

```
[
  {
    "type": "doc",
    "document": "Jamba: A Hybrid Transformer-Mamba Language
       ↪ Model\n Opher Lieber, Barak Lenz, ..."
  },
  {
    "type": "qna",
    "question": "Who are the authors of the Jamba paper??",
    "answer": "The authors of the Jamba paper are Opher Lieber,
       ↪  Barak Lenz, ..."
  },
  {
    "type": "qna",
    "question": "What is Jamba and what architecture does it
       ↪ utilize?",
    "answer": "Jamba is a novel architecture utilizing
       ↪ Transformer and Mamba layers..."
  },
]
```

Interleaving text and question-answer pairs For each chunk of raw text fed into the model, we augmented it with 3–5 open-ended question–answer pairs specific to that chunk. With the raw text,

the idea was that the model would continue to draw relations between the tokens and learn knowledge, and with the question–answer pairs, we explored the possibility of a more focused factual extraction coupled with the maintenance of its instruct-tuned nature.

```
def formatting_prompts_func(examples):
    types        = examples["type"]
    documents    = examples["document"]
    inputs       = examples["question"]
    outputs      = examples["answer"]
    texts = []
    for example_type, document, input, output in zip(types,
        ↪ documents, inputs, outputs):
        if example_type == "doc":
            text = document + EOS_TOKEN
        else: # example_type == "qna"
            text = alpaca_prompt.format(input, "", output) +
                ↪ EOS_TOKEN
        texts.append(text)
    return { "text" : texts, }
```

Note that strictly speaking, we are performing autoregressive language modeling, where the distinction between supervised and unsupervised fine-tuning for LMs starts to blur. In our case, question–answer pairs are mapped to text chunks and interleaved with raw text from the training documents.

4.2 Implementation

Code copilot: First, we chose as our specialized dataset a Python repository created in January 2024, ensuring its absence from any public training set[2]. We created a library that recursively traverses each file and directory, splitting each `.py` file into 4KB-sized chunks. We used LangChain's text-splitter library [3] to ensure that code chunks ended in appropriate areas, e.g. the end of a logical code chunk.

Research assistant: Reusing the aforementioned library, we chose 10 research papers published no earlier than November 2023, split them into chunks, and generated question-answer pairs.

Enterprise chatbot: Finally, we took product titles and descriptions in the `wdc/products-2017` database[4] and performed some data processing to remove some metadata and other irrelevant text, obtaining clean chunks which emulate company product offerings.

RAG support: Additionally, off-the-shelf evaluation methods do not support RAG, so we built a new library that appends the dataset with an additional *context* column that provides relevant context during testing. This *context* represents the top-k documents that would have been returned by a retriever.

The collected chunks were then fed into OpenAI's `gpt-4-turbo` model, which generated question-answer pairs in the style shown above. These were then appended to the original dataset containing the raw chunks. Finally, all datasets were pushed to Huggingface for reference.

4.3 Training Details

Hyperparameters: We start with instruct-tuned versions of both Mamba-2.8b and Gemma-2.5b. Using the Huggingface Supervised Fine-Tuning Trainer [5], each model was fine-tuned on 3 epochs with a learning rate of 2×10^{-4}, the same learning rate used for instruct-tuning. We used the Adam optimizer with a linear learning rate scheduler and set a max sequence length of 2048 tokens.

Hardware: Both Mamba-2.8b and Gemma-2.5b were trained on a single NVIDIA A100_80G GPU.

RAG: We used LangChain's embeddings library for RAG, embedding our custom datasets directly from HuggingFace to ChromaDB, an open-source vector store that works well for embedding and

[2] `https://github.com/scottsus/flamethrower`

[3] `https://python.langchain.com/docs/modules/data_connection/document_transformers`

[4] `https://huggingface.co/datasets/wdc/products-2017`

[5] `https://huggingface.co/docs/trl/en/sft_trainer#trl.SFTTrainer`

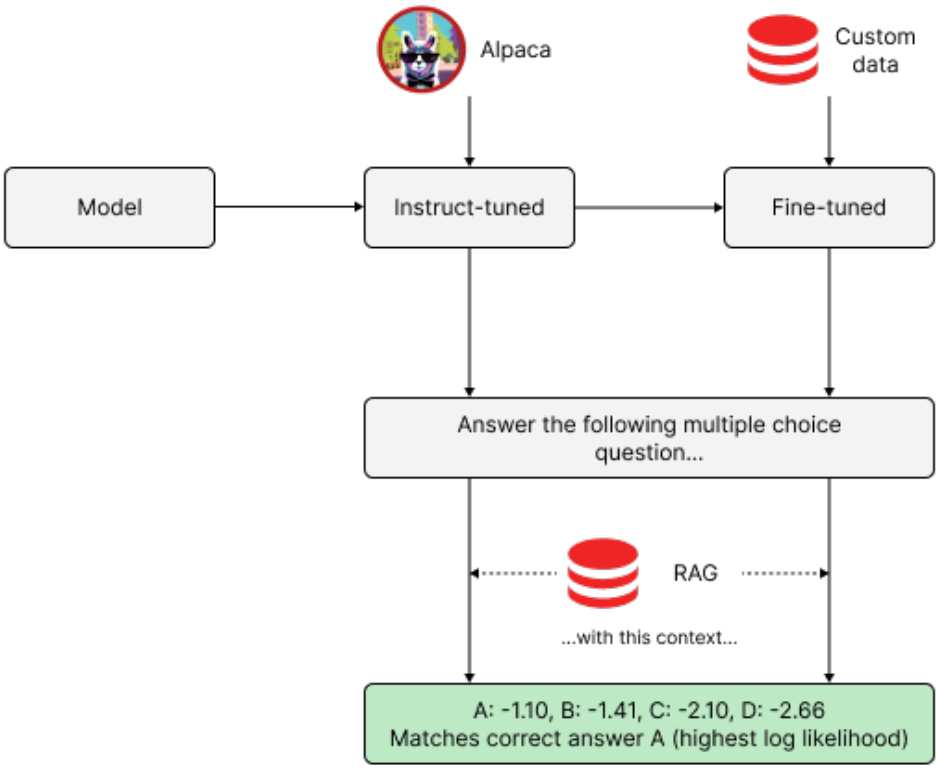

Figure 1: Architecture diagram

retrieving documents. We used `BAAI/bge-large-en-v1.5`[6] as the embedding model for its solid performance on the HuggingFace Massive Text Embedding Benchmark (MTEB) leaderboard and low resource demand.

5 Experiments

5.1 Evaluation

We evaluated the performance of our LMs using `EleutherAI/lm-evaluation-harness` [7], a SOTA framework that simplifies and standardizes the evaluation of language models across many different tasks including text generation, text summarization, named entity recognition, and in our case, question answering for multiple-choice questions.

In our multiple-choice setting, `lm-evaluation-harness` creates four strings for each question, each of the form

```
{question}
A: {choice[0]}
B: {choice[1]}
C: {choice[2]}
D: {choice[3]}
Answer: {A, B, C, or D}
```

The harness then presents the LM with each of the four strings, each representing an "alternate reality" where one of the four choices is correct, and asks the LM to evaluate the statistical likelihood

[6] `https://huggingface.co/BAAI/bge-large-en-v1.5`
[7] `https://github.com/EleutherAI/lm-evaluation-harness`

of each string per its weights. The harness obtains a list of four log-likelihoods; the model is taken to have "selected" whichever answer it said had the greatest log-likelihood. In the end, its accuracy is evaluated as the count of questions it correctly answered divided by the total number of questions.

Soft lower and upper bounds: Previously, we mentioned that a model is said to have learned new facts if it obtained more correct answers for the unseen data as compared to the version of it that was not fine-tuned. This sets a soft lower bound for model performance. On the other hand, RAG is an industry standard, off-the-shelf solution for modern LM use cases, so it sets a soft upper bound for performance.

Test set: Our evaluation datasets were generated using OpenAI's `gpt-4-turbo`. Conversely to generating open-ended question-answer pairs for unsupervised fine-tuning, we also used `gpt-4-turbo` to generate multiple-choice question-answer pairs for evaluation using a subset of the initial chunks used. Below was the system message we fed into `gpt-4-turbo` to generate the evaluation questions.

> You are a world-class university professor constructing an exam.
> Given a piece of raw text, produce `questions`, a list of question and answer pairs of size 3-5.
> Each question and answer pair contains:
> - a single question
> - 4 possible choices (A, B, C, D) of which only 1 is correct
> - a correct answer -> this should be *only a single letter: the correct option*.
> A JSON schema is provided for your reference: {JSON_SCHEMA}

Using this method we generated a total of 4,628 new questions for testing.

Table 1: Evaluation questions generated for each task

Task	**Eval size**
Code	1,974
Research	263
Products	2,391

Conversational behavior: Manual evaluation using regular question answering was done for each model before and after fine-tuning to check for the retention of assistant-like behavior following the Alpaca format.

5.2 Results

After conducting the experiments, we observed mixed results from the 2 models in their capabilities for learning new information after fine-tuning. Mamba-2.8b appears to benefit from fine-tuning, successfully learning new factual knowledge for certain tasks, whereas Gemma-2.5b appears to suffer from new, unseen data, at some points even regressing and performing worse than before it was fine-tuned. Recall that it is expected for RAG to outperform regular fine-tuning in all tasks.

Code: Requiring very specific, real-time context with the most recent updates to the repository, it is unsurprising that a fine-tuned model would perform at a similar level to the base version. This was true for Mamba-2.8b, which improved by about +2.66%. However in Gemma-2.5b, fine-tuning might have worsened its generalization capabilities, and if at first it was able to make some educated guesses about code-related questions, it was later unable to match the same performance after fine-tuning.

Research: Learning and (even more difficult, cognitively speaking) *understanding* deep academia is expectedly a difficult task for small LMs. Again, looking at the most significant delta in performance, we note that Gemma-2.5b was able to make some educated guesses during eval, but after fine-tuning, it lost significant generalization abilities and ended up performing worse than before. By the same token, Mamba-2.8b did not do so well with novel and advanced research concepts, but it suffered a significantly smaller dent in performance.

Products: For this task, because of the minimal context needed to answer questions, we hypothesized that fine-tuning would stand the greatest chance of improving the performance of the models. The

+3.56% increase in performance from Mamba-2.8b lends strong evidence to our hypothesis for this task. However, for a third time, Gemma-2.5b degraded in performance after fine-tuning.

5.3 Analysis

Gemma's proprietary training: Gemma's degradation after our fine-tuning process lends credibility to the suspicion that Gemma-2.5b was instruct-tuned on some high-quality, proprietary training heuristics that may enabled it to make some educated guesses on unseen data. However, our custom fine-tuning process may have interfered with Gemma's training heuristics, causing it to regress and perform badly after fine-tuning.

Positive results for Mamba: This Mamba-2.8b model checkpoint was instruct-tuned on the `yahma/alpaca-cleaned` dataset[8] without additional training heuristics or prompt engineering besides the Alpaca format, and shows significant improvement during fine-tuning on unseen data, confirming our hypothesis that instruct-tuned models can indeed learn new information without losing their instruct-behavior.

Table 2: Results for datasets described in terms of log-likelihood accuracy

Task	Model	Base model	Fine-tuned	RAG	Fine-tuned + RAG
Code	Mamba-2.8b	0.2586	**0.2852**	0.2776	0.2700
	Gemma-2.5b	0.3764	0.2877	**0.4295**	0.2281
Research	Mamba-2.8b	0.3117	0.3072	**0.3315**	0.3167
	Gemma-2.5b	**0.3674**	0.1888	0.2961	0.1923
Products	Mamba-2.8b	0.3191	0.3547	**0.3572**	0.3527
	Gemma-2.5b	0.2877	0.2200	**0.4918**	0.1924

5.4 Rejecting the Null Hypothesis

Although the improvement for Mamba-2.8b is relatively small, we showed that instruct-tuned models can in fact continue to learn factual knowledge using our novel method in the face of new information.

6 Conclusion and Future Work

Large language models are often thought of as lossy compressions of a curated internet of text, and their ability to understand, memorize, and recall information provides significant utility to our modern society in a variety of fields, such as research, medicine, enterprise, and engineering. In this work, we test the ability of instruct-tuned LMs to learn new knowledge without losing their ability to interface with end-users with assistant-like behavior.

Recap on results We explore a novel method of interleaving raw text (like those seen during pre-training) and question-answer pairs (like those seen during instruct-tuning) in an attempt to teach the model new knowledge while retaining its instruct-behavior. While performance degraded for Gemma-2.5b, our results showed that performance increased for Mamba-2.8b, proving that, although not incredibly significant at +3.56%, it is in fact possible for instruct-tuned models to learn new things while retaining their conversational behavior.

Sutton's Bitter Lesson In this work we explored only models with about 3B parameters. In Sutton's article *The Bitter Lesson* [10], he outlined that scaling laws dominate all human-centered heuristics and by a large margin, they potentially nullify our human and marginal efforts to improve the models' performances on unseen data. We strongly recommend trying this method with larger models to observe better generalization in the face of new information.

Highly performant compact models In addition to scaling up, very recent advancements in dataset quality and hyperparameter optimization have enabled the existence of medium-sized LMs that have performance on par with very large LMs [6, 3], and there is a possibility that heuristics

[8] `https://huggingface.co/datasets/yahma/alpaca-cleaned`

learned during pre-training may be beneficial for further learning during later phases of supervised and unsupervised fine-tuning.

Mixture-of-Experts Most recently, developments in the Mixture-of-Experts (MoE) architecture have achieved incredible performance while utilizing only a fraction of its total parameters for inference [8, 2]. It works by having a number of different *experts* whose activation is chosen by a *router*, both of which are learnable parameters. A natural continuation of our work then becomes *Could MoE models learn new things?*

7 Limitations

As with all machine learning projects, the choice of hyperparameters significantly impacts the end result. In this work we stuck with a fixed set of hyperparameters, including learning rate equal to the instruct-tuning learning rate, number of epochs set to the default of 3, and several other hyperparameter choices. We recommend performing a hyperparameter search to find the best set of hyperparameters for these and other types of models.

Mamba, the most popular SSM and open-sourced by the authors themselves, has a ceiling of 2.8B parameters, and thus we were unable to obtain fair comparisons of Mamba vs Gemma for models comparable in size to 7B parameters. We recommend exploring scaling up Mamba to at least 7B parameters for future experiments.

References

[1] Tri Dao Albert Gu. Mamba: Linear-time sequence modeling with selective state spaces. 2023.

[2] Antoine Roux Arthur Mensch Blanche Savary Chris Bamford Devendra Singh Chaplot Diego de las Casas Emma Bou Hanna Florian Bressand Gianna Lengyel Guillaume Bour Guillaume Lample Lélio Renard Lavaud Lucile Saulnier Marie-Anne Lachaux Pierre Stock Sandeep Subramanian Sophia Yang Szymon Antoniak Teven Le Scao Théophile Gervet Thibaut Lavril Thomas Wang Timothée Lacroix William El Sayed Albert Q. Jiang, Alexandre Sablayrolles. Mixtral of experts. 2024.

[3] Misha Bilenko. Introducing phi-3: Redefining what's possible with slms. 2024.

[4] Kang Min Yoo Bruce W. Lee, Hyunsoo Cho. Instruction tuning with human curriculum. *University of Pennsylvania, Seoul National University*, 2024.

[5] Adam Ibrahim Mats L. Richter Quentin Anthony Eugene Belilovsky Irina Rish Timothee Lesort Kshitij Gupta, Benjamin Therien. Continual pre-training of large language models: How to (re)warm your model? 2023.

[6] Meta. Introducing meta llama 3: The most capable openly available llm to date. 2024.

[7] Moshik Mishaeli Oren Elisha Oded Ovadia, Menachem Brief. Fine-tuning or retrieval? comparing knowledge injection in llms. 2023.

[8] Hofit Bata Gal Cohen Jhonathan Osin Itay Dalmedigos Erez Safahi Shaked Meirom Yonatan Belinkov Shai Shalev-Shwartz Omri Abend Raz Alon Tomer Asida Amir Bergman Roman Glozman Michael Gokhman Avashalom Manevich Nir Ratner Noam Rozen Erez Shwartz Mor Zusman Yoav Shoham Opher Lieber, Barak Lenz. Jamba: A hybrid transformer-mamba language model. 2024.

[9] Srikanth Thudumu Zach Brannelly Mohamed Abdelrazek Scott Barnett, Stefanus Kurniawan. Seven failure points when engineering a retrieval augmented generation system. *Applied Aritifical Intelligence Institute*, 2024.

[10] Rich Sutton. The bitter lesson. 2019.