On Imitating Proprietary Language Models

Arnav Gudibande
Dawn Song, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-149
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-149.html

May 12, 2023

On Imitating Proprietary Language Models

Arnav Gudibande

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee

Dawn Song

Dawn Song
Research Advisor

5/9/2023

(Date)

★ ★ ★ ★ ★ ★ ★

DocuSigned by:
Dan Klein
8862E77FBEC3465...

Dan Klein
Second Reader

5/10/2023

(Date)

Abstract

Fine-tuned language models (LMs) provide the backbone for popular services such as ChatGPT, GitHub Copilot, and Cohere AI. The competitive edge of these systems often arises from their proprietary finetuning data (e.g., user-submitted prompts), and thus companies invest substantial resources into collecting and protecting this data. In this work, we study model "imitation" as a method to close the gap between open-source LMs and their closed-source counterparts. In the first part, we propose a framework for cheaply imitating proprietary language models in specific domains. In particular, we create a prompting pipeline that first asks what tasks a particular LM can solve and then asks for input-output examples for those tasks. We then fine-tune open-source LMs on these supervised input-output examples to create imitation models. We show that human evaluators rate the outputs of these imitation models more highly as these models get larger and use bigger querying budgets. In the second part, we apply this general framework to ChatGPT and release Koala, our strongest imitation model. Initial evaluations show that this model results in impressive qualitative performance compared to ChatGPT in specific domains.

Acknowledgement

I would like to thank my research mentors Xinyun Chen and Eric Wallace for taking me on as a student and providing invaluable day-to-day mentorship. I'd also like to thank my advisors Dawn Song and Dan Klein for supporting my work and giving high quality feedback. This work would not have been possible without my collaborators Charlie Snell, Xinyang Geng, Hao Liu and others in the Berkeley NLP Group and BAIR. Finally, I would like to thank my parents for their continued support and love throughout this journey.

A Framework for Imitating Proprietary Language Models

Arnav Gudibande [1] **Eric Wallace** [1] **Dawn Song** [1]

Abstract

Fine-tuned language models (LMs) provide the backbone for popular services such as ChatGPT, GitHub Copilot, and Cohere AI. The competitive edge of these systems often arises from their proprietary finetuning data (e.g., user-submitted prompts), and thus companies invest substantial resources into collecting and protecting this data. In this work, we demonstrate a framework for imitating proprietary language models in specific domains. In particular, we create a prompting pipeline that first asks what tasks a particular LM can solve and then asks for input-output examples for those tasks. We then finetune open-source LMs on these supervised input-output examples to create imitation models. As a case study, we use this pipeline to imitate Instruct-GPT (text-davinci-002), where by using $500 of API queries we reach ∼93% of its performance according to human evaluation on a specific distribution of broad-coverage tasks. Overall, we demonstrate that proprietary models like Instruct-GPT can be used to train imitation models that are of generally high qualitative performance in specific domain distributions for a relatively cheap cost.

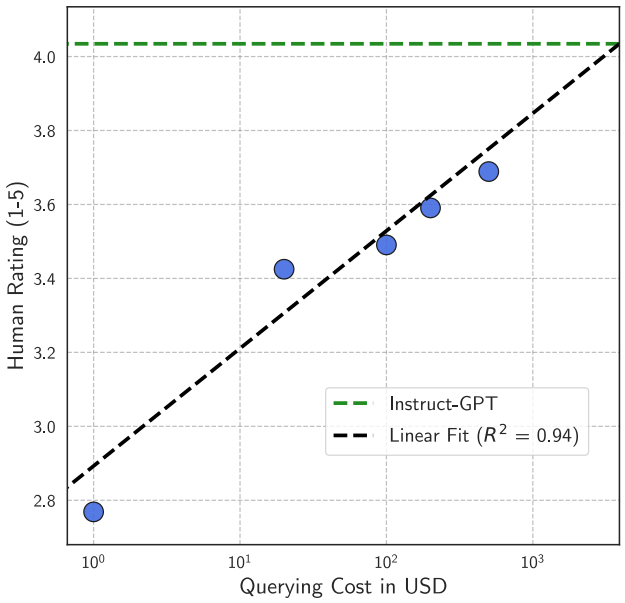


Figure 1. We finetune a 6B parameter LM to imitate the outputs of Instruct-GPT using varying amounts of API queries on a specific distribution of task instructions. As our budget increases, human evaluators rate the outputs of our imitation LMs to be higher quality, following a log-linear relationship. We spend $500 to achieve ∼93% of Instruct-GPT's performance—we estimate that $2500 worth of queries would be needed to fully match its performance on this particular evaluation distribution.

1. Introduction

Large language models (LMs) can solve numerous downstream tasks when prompted with natural language instructions (Brown et al., 2020). Recently, instruction-tuned LM variants such as Instruct-GPT (Ouyang et al., 2022), ChatGPT (OpenAI, 2022), and Codex (Chen et al., 2021) have dramatically improved LM capabilities by fine-tuning models on domain-specific datasets. These models are lucrative assets—some are served behind paid APIs with millions of users (Metz & Weise, 2023)—and are built using proprietary datasets and algorithms. Consequently, many companies keep their model weights private to protect intellectual property and maintain their competitive edge.

In this work, we propose a framework where proprietary LMs can be imitated in specific domains by simply interacting with their APIs. We consider a setting where one collects outputs from the API and uses them to fine-tune their own *imitation* LM. We show that model imitation can successful because proprietary LMs are capable of generating high quality input and output examples. Therefore, one can create a large dataset to imitate the overall performance of these proprietary systems in particular domains.

As a case study, we imitate Instruct-GPT (text-davinci-002), a widely-used instruction-tuned LM that is finetuned on proprietary data to solve zero-shot tasks. We query the target LM using a pipeline where we first prompt it to output a list of broad-coverage task instructions. We then feed each one of these task descriptions to the LM and ask it to synthesize associated input-output examples. We collect datasets using

[1]UC Berkeley. Correspondence to: Arnav Gudibande <arnavg@berkeley.edu>.

this pipeline that range from 10K examples (∼$20 USD) to 250K examples (∼$500 USD) [1].

Using these datasets, we finetune open-source LMs across a range of sizes (125M–6B). We evaluate them using human preference ratings on broad-coverage instructions, as well as automated zero-shot evaluations on public test sets (Super-NI, Wang et al. 2022b). Our imitation models quickly approach the quality of Instruct-GPT in this domain. For example, human evaluators rate our system trained with $500 worth of queries an average of 3.7 out of 5, or ∼93% as good as Instruct-GPT which had an average rating of 4. Moreover, using larger query budgets can likely further close this gap—in Figure 1, we fit log-linear trends that estimate only ∼$2500 worth of queries would be needed to fully match Instruct-GPT's performance on this particular domain distribution and evaluation setting.

We stop short of claiming that this method can lead to broad imitation of proprietary language models across all domains, since the querying costs and amount of data needed would likely be unwieldy. However, our work shows that proprietary language models like Instruct-GPT can be used to train imitation models that are of generally high qualitative performance in a specific domain distribution for a relatively cheap cost.

2. Background and Setup

Instruction-tuned Language Models Many recent efforts have improved the zero-shot abilities of LMs via "instruction" or "meta" finetuning (Ouyang et al., 2022; Min et al., 2022; Zhong et al., 2021; Wei et al., 2022, *inter alia*). Such instruction-tuned LMs have become extremely popular and lucrative, e.g., InstructGPT has millions of users (Metz & Weise, 2023), and there is thus competition among companies to continually improve these models. To do so, organizations invest substantial resources into building proprietary instruction-finetuning datasets (Ouyang et al., 2022) which have become a key component that differentiates the LMs and API services of various companies.

Stealing NLP Models Proprietary LMs are served behind black-box APIs that allow users to generate outputs for arbitrary inputs. API users may otherwise be unaware of the model's next-token probabilities, its architecture, hyperparameters, or training data. In our paper, we consider an adversary whose goal is to use this API access to train an *imitation model* (Orekondy et al., 2019; Wallace et al., 2020) that achieves comparable performance to the proprietary LM on held-out data. The adversary can then use this model for financial gain, e.g., to launch their own competitor service or to avoid paying for future API queries.

[1] Based on OpenAI API Pricing from December 2022

Prior work has shown that model stealing[2] is possible for various domains (Lowd & Meek, 2005; Tramèr et al., 2016; Orekondy et al., 2019), including language classifiers (Krishna et al., 2020; Pal et al., 2019) and machine translation systems (Wallace et al., 2020). Nevertheless, past work imitates models that solve *one* task and are not built using large LMs. We instead focus on imitating models that are massively multi-task in nature and are fine-tuned from large LM checkpoints. In the subsequent section, we discuss the challenges and opportunities that arise from our setting.

3. Model Imitation Method

Perhaps the key challenge for imitating proprietary models is to decide what inputs to use when querying its API for training data. Past work assumes that the adversary has access to a pool of inputs (e.g., Wikipedia sentences) and must decide which ones to query (Orekondy et al., 2019; Wallace et al., 2020). In our case, we *do not have access to inputs to query*. Our inputs are natural language instructions for arbitrary tasks, e.g., "Write a poem about your favorite season", which are not readily available.[3]

Our key technical contribution is to show that state-of-the-art large LMs can synthesize high-quality *inputs* and outputs. This dramatically simplifies model stealing as we can simply prompt models to generate synthetic data to use for training, rather than carefully choosing inputs to query. In fact, because large LMs are increasingly adept at memorizing their training data (Carlini et al., 2021a; Kandpal et al., 2022), we may even expect that these synthetic datasets resemble a company's internal proprietary dataset.

In our work, we focus on instruction-tuned LMs (although our methods are general), where the "inputs" consist of both task instructions and optional input text, and the "outputs" are the corresponding model predictions. To generate data for our imitation LMs, we create a two-stage pipeline (Figure 2). The first stage asks the target LM to generate a list of tasks that it can solve. The second stage then generates input and output examples for each of those tasks. We then create our imitation models by fine-tuning open-source LMs on these task instructions + input-output examples.

3.1. Collecting Task Instructions

In the first stage of our pipeline, we collect a large list of task instructions (i.e., textual definitions and descriptions of

[2] Tramèr et al. (2016) popularized the term "model stealing" to describe these types of attacks. We instead tend towards the term "model imitation" to clearly indicate that we are not directly extracting the target model or reverse engineering its weights.

[3] Note that our goal is to describe a method to imitate *arbitrary* proprietary LMs. However, our experiments focus on imitating *instruction-tuned* models, where there are open-source datasets. For scientific purposes, we do not assume any access to this data.

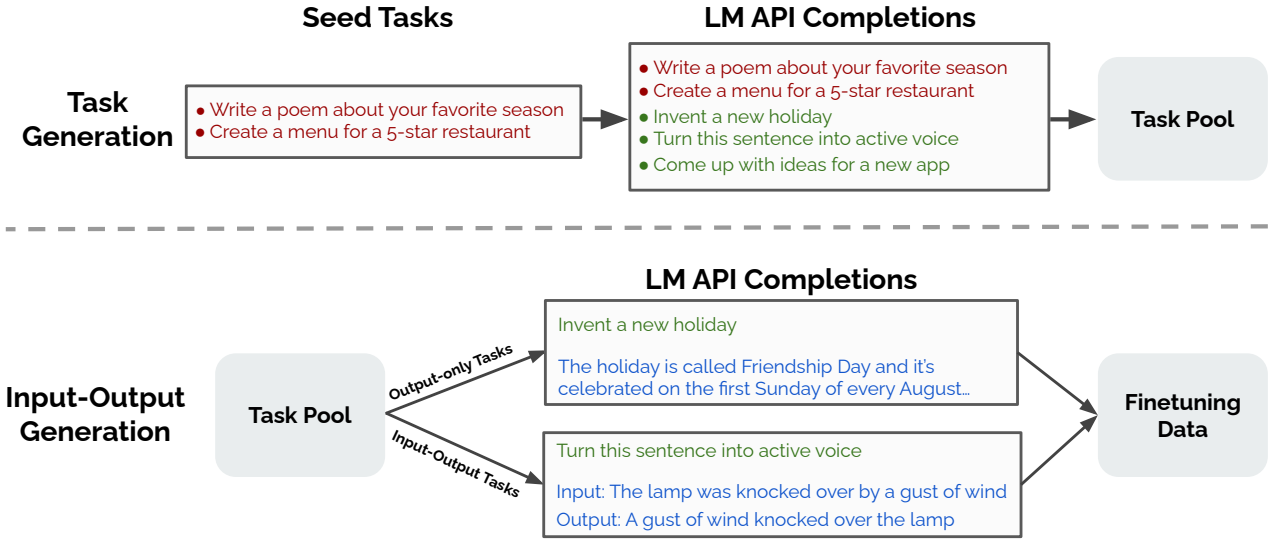


Figure 2. Our data collection pipeline is made up of two parts, generating task instructions (*top*) and then generating input-output instances (*bottom*) for each task instruction. To generate task instructions, we prompt the target LM with a list of seed tasks (marked in red), then we store the generated task instructions (marked in green) in a task pool. Once we have collected a task pool, we prompt the target LM to generate either a single output or both an input and output for every task instruction (marked in blue).

tasks) from the target LM. Task instructions may require generating open-ended text such as "Write a poem about your favorite season" or "Design a new type of clothing". We refer to these open-ended tasks as *output-only*. Instructions also may include references to an unspecified text input, e.g., "Group this list of names into male and female" or "This sentence is unclear, rewrite it using different words". We refer to this style of tasks as *input-output* tasks.

Bootstrapped Instruction Generation The main challenge in synthesizing task instructions is to ensure that they are large in quantity while maintaining diversity. To accomplish this, we first manually write a set of ten task instructions that are representative of a particular task category (shown in Table 3 in Appendix B). We craft these instructions to span different categories and to sometimes refer to unspecified inputs. Given this seed set of instructions, we then generate thousands of novel instructions from the target LM.

To do so, we prompt the LM with a bulleted list of three task instructions and generate from the model with top-p sampling ($p = 0.8$) until it generates an end-of-text character. We then sample a fresh set of three task instructions and repeat this process. To sample the three instructions, we choose from a pool of instructions that is grown in a bootstrapped fashion. We initialize the pool with the ten seed instructions for a particular category and then add each subsequent model generation to the pool. Consequently, the pool transitions from containing only manual instructions to eventually consisting of predominately model-generated instructions. Moreover, we further encourage diversity in the pool of examples by not adding any instruction to the pool if it has greater than 20% BLEU-score similarity with any existing example in the pool. We repeat this overall process for each category of task. See Appendix B for further details on our post-processing, prompts, and seed tasks.

3.2. Generating Input-Output Examples

Now that we have generated a large set of instructions, we next turn to generating corresponding inputs and outputs. As mentioned, task instructions can refer to output-only or input-output tasks. We know which type an instruction falls in based on the task category it came from (e.g., classification tasks are input-output).

For output-only tasks, we query the target LM to generate an output, e.g., for the task "Design a new type of clothing" the output may be "I'm thinking a dress that is made to look like a two piece, but is actually one piece...". We provide the target LM with the instruction and generate with top-p sampling ($p = 0.9$).

For input-output tasks, we need the LM to synthesize an input to process and a corresponding output. These instructions can have many formats. In contrast to other work on instruction-tuning that uses fixed input and output templates (Wang et al., 2022b), we do not make any assumptions on the templates and instead allow the target LM to generate any format it desires. We append phrases such as "generate an output for the following input" to the instructions to encourage the LM to generate both an input and an output. The exact phrase that we append depends on the task category; Appendix B.1 details all phrases used. We then generate

Category	Instruct-GPT Generation
Text Generation	Create a character and come up with their back story, family history, and personality traits. Name: Wren Back story: Wren was abandoned by her parents as a baby and raised in the forest by a family of birds. She has never known any other life than the one she has in the forest and is completely content with that. She is a very independent and self-sufficient and is also very compassionate and caring towards all of the... Personality traits: independent, self-sufficient, compassionate, caring
Brainstorming	Write a list of random acts of kindness you can do for others. • Buy a stranger a cup of coffee • Pay for the person behind you in line • Leave a nice note for someone • Do someone's grocery shopping for them • Mow a neighbor's lawn
Text Extraction	Find all the dogs in this list that are over 2 years old. Input: • Chihuahua, 3 years old • Pomeranian, 1 year old • Labrador Retriever, 2 years old • Golden Retriever, 4 years old Output: • Chihuahua, 3 years old • Golden Retriever, 4 years old

Figure 3. Examples of task instructions and outputs that are generated by Instruct-GPT and used in our imitation LM dataset. Overall, we find that instruction-tuned LMs are capable of generating high-quality and diverse prompt instructions that span multiple categories such as open-ended text generation, brainstorming, and extraction. Moreover, for categories such as text extraction, we find that LMs are surprisingly capable of producing high-quality synthetic *inputs* and corresponding outputs.

with this full prompt using top-p sampling ($p = 0.9$).

3.3. Imitation Model Training

After collecting the model outputs, we concatenate them to the original task instructions. We then finetune open-source LMs on this data using standard autoregressive training objectives. Our models do not make any explicit disambiguation between the instructions and the inputs or outputs.

4. Imitating Instruct-GPT

As a case study, we apply our model imitation pipeline to Instruct-GPT (Ouyang et al., 2022), a popular instruction-tuned LM available through the OpenAI API. We target text-davinci-002 as it was the best model available when we conducted our experiments. This model is finetuned using proprietary data consisting of annotated task instructions and outputs (Ouyang et al., 2022). We emphasize that our method is not specific to OpenAI's API and our aim is not to exploit OpenAI or its products.

4.1. Imitation Dataset

Ouyang et al. (2022) discuss the broad capabilities of Instruct-GPT across various task categories. We aim to imitate the model across a specific range of categories,

Quality Review Question	Yes %
Is the task instruction valid?	97%
For input-output tasks, is the input appropriate for the instruction?	71%
Given a valid instruction and input, is the output appropriate?	80%
For output-only tasks, is the output appropriate for the instruction?	93%
All fields are valid	83%

Table 1. We conduct a manual quality review of 100 random instructions, and the corresponding output-only or input-output instances. The dataset is of high-quality, e.g., nearly all of the synthetic task instructions are valid and 93% of the outputs for output-only tasks are of good quality.

which we group as: *output-only*—text generation, brainstorming, chat, and open-ended QA—and *input-output*—text rewriting, extraction, classification, and closed-ended QA. Instruct-GPT is trained to capture a specific distribution over these task categories (see Table 2). We aim to imitate the model across this same distribution, but we note that adversaries can generate task instructions according to arbitrary distributions based on their intended use case. We provide more details on each task category in Appendix A.

We collect datasets consisting of 200, 1000, 2000, and 5000 unique task instructions. We then query the API to collect 50 unique outputs (and possibly synthetic inputs) for each instruction. This yields final training sets of 10K, 50K, 100K, and 250K instruction and example pairs. The total cost of collecting this data is approximately $20, $100, $200, and $500 respectively.[4]

Dataset Quality We find that our imitation data is surprisingly high quality given its automatic construction. First, there is high diversity in the instructions. We find that at the example level, the average BLEU score similarity among instances is 18%. This is considerably lower than similar manually-written datasets from the literature, e.g., this value is 61% for a comparably sized subset of Super-NaturalInstructions (Wang et al., 2022b).

We also conducted a manual review of a random subset of 100 examples from our 100K dataset. For each example, we evaluated (1) whether the instruction was valid and grammatical, (2) if there were generated inputs, were they appropriate for the instruction, and (3) whether the outputs were correct and appropriate. We show the results in Table 1. The data is surprisingly high-quality: nearly all of the task instructions are valid and a vast majority of the synthetic instances contain valid input-output pairs.

4.2. Training and Evaluating Imitation Models

We consider open-source decoder-only LMs that range from 125M–6B parameters:

- **GPT-2 Small**: We use the 125M GPT-2 model trained by OpenAI on WebText (Radford et al., 2019).
- **GPT-2 XL**: We use the 1.5B GPT-2 model.
- **GPT-J**: We use GPT-J, a 6B parameter LM trained by EleutherAI on The Pile (Black et al., 2022).

We fine-tune these models on the various synthetic datasets (10K, 50K, 100K, and 250K) examples. This allows us to study the impact of both dataset scaling and model scaling on imitation model results. We fine-tune autoregressively for one epoch using the AdamW optimizer with learning rate 1e-5 and batch size 16. All models are trained using distributed JAX on TPUs hosted by Google Cloud.

Evaluation Setup We employ two evaluations: human evaluation on a set of held-out instructions and automatic evaluation on Super-NaturalInstructions (Wang et al., 2022b, Super-NI). For all experiments, we compare our imitation LMs against Instruct-GPT and a pre-trained GPT-J baseline.

For human evaluation, we ask raters on Amazon Mechanical Turk to score the outputs of a model from 1–5 (in terms of quality and correctness, 5 is higher) on a set of held-out instructions. In our UI, we present each rater with a task instruction, optional input, and the corresponding model output (see Figure 5 in Appendix B.2). We conduct our evaluation on 500 held-out instructions. For those instructions that are input-output tasks, we use human-written inputs to ensure that they are valid. This is also collected using Mechanical Turk, see Figure 6 in Appendix B.2.

We also consider automatic evaluation using Super-NaturalInstructions (Super-NI). We use SuperNI in the zero-shot setting, where models are prompted with a task definition without any demonstration outputs. Following Wang et al. (2022b), we measure the Rouge-L score for each model's output with the demonstration output. We caveat that although SuperNI is a large benchmark for evaluating zero-shot performance, it comes from a different distribution than the data that we query to the API. In fact, the Instruct-GPT paper notes that as they collect and fine-tune on larger internal datasets, their model actually gets worse on public benchmarks (Ouyang et al., 2022). Therefore, we do not expect our models that imitate Instruct-GPT to perform particularly well on SuperNI. Nevertheless, we used SuperNI as an automated signal during our preliminary experiments and we report results on it for completeness.

4.3. Key Results

Human Evaluations Our imitation models substantially close the gap to Instruct-GPT. First, we plot the average human rating for our fine-tuned GPT-J models in Figure 1. As the querying budget increases, human evaluators tend to rate the outputs of our imitation models higher, following a log-linear relationship. We fit scaling laws (line of best fit assuming log-linear trend) to this data, which suggests that approximately $2500 worth of queries would be needed to match the performance of Instruct-GPT on our evaluation dataset. Given that we had a limited budget remaining, we leave collecting this larger dataset to future work.

In Table 2, we report the average human ratings broken down by task category. Our imitation GPT-J trained on 250K samples most closely matches the quality of Instruct-GPT on tasks such as text generation, brainstorming, chat, and question-answering. Overall, with just $500 of queries, our imitation model approaches approximately $\sim$93% of the average quality of Instruct-GPT across all tasks.

Automated Evaluations We next evaluate our imitation models on SuperNI and report the average Rouge-L score in Table 9 in Appendix C. The 10K and 50K models demon-

[4]Based on the OpenAI API costs from December 2022. We calculate these costs based on the average token length of our instructions and model generations. These numbers may deviate slightly depending on the exact queries made.

Category	Frequency	Imitation Dataset Size					
		GPT-J	10K	50K	100K	250K	Instruct-GPT
Generation	46.0	2.9	3.6	3.6	3.6	3.8	4.0
Open QA	12.8	3.0	3.7	3.5	3.7	3.8	4.3
Brainstorming	11.6	3.2	4.1	4.0	4.2	4.1	4.3
Chat	8.8	2.7	3.6	3.5	3.7	3.8	4.1
Rewrite	7.0	2.1	2.3	2.9	2.8	3.1	3.8
Summarization	4.4	2.3	2.6	2.9	3.3	3.3	3.9
Classification	3.9	1.8	1.5	2.5	2.4	2.2	3.8
Closed QA	3.0	2.6	4.1	4.0	4.3	4.5	4.3
Extract	2.3	1.8	1.8	2.2	2.6	2.3	3.8
Weighted Avg	100	2.8	3.4	3.5	3.6	3.7	4.0

Table 2. We finetune GPT-J on increasing amounts of imitation data and compare our imitation models to a non-finetuned GPT-J and Instruct-GPT using human evaluations from 1–5. With $500 of queries (250K data points), our imitation models significantly close the gap to Instruct-GPT across many categories, even sometimes slightly outperforming it. Our log-linear scaling trends predict that with a higher querying budget or larger pre-trained models, we can fully close the gap to Instruct-GPT on this evaluation setting.

strate significant improvements in the average Rouge-L score over the pre-trained GPT-J model. However, we note that further finetuning the model on the 100K and 250K datasets actually decreases performance on SuperNI. This aligns with findings from Ouyang et al. (2022), who mention that their instruction-tuned models get worse according to public benchmarks when they are further fine-tuned on domain-specific task instructions. Overall, SuperNI provides another signal that we drastically improve over the initial GPT-J model using our imitation data, but we consider the human evaluation results to be of higher relevance and importance.

Scaling Model Size Helps Increase Performance Imitating proprietary LMs hinges on the availability of open-source pre-trained LMs that are somewhat comparable in capabilities to the target model. In our case, we imitate text-davinci-002, which is based on the 175B parameter GPT-3 pre-trained model. Part of the gap from our imitation model to Instruct-GPT can therefore be attributed to us using the relatively small 6B parameter GPT-J LM.

To explore this further, we study the effect of model scaling using our 50K dataset. We fine-tune GPT-2 small, GPT-2 XL, and GPT-J on the data and plot their average human ratings in Figure 4. With a fixed query budget, scaling the size of the models improves average human ratings. The scaling trend shows a moderate correlation, and we could thus likely further close the gap to Instruct-GPT by using larger and more powerful open-source models, although it is unclear as to exactly how much. We leave this experiment to future work due to computational constraints.

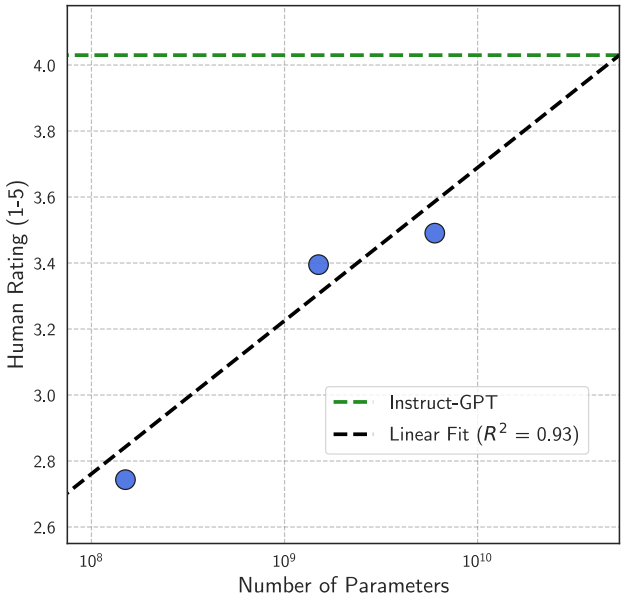


Figure 4. Model scaling. We finetune GPT2 small, GPT2-XL, and GPT-J models on our 50K imitation dataset and ask humans to rate the output quality. Scaling the size of the base LM increases the quality of the imitation model. We also fit scaling laws that show moderate correlations and suggest that we can further close the gap to Instruct-GPT by using larger open-source models.

5. Discussion and Related Work

Model Distillation Methodologically, model imitation is related to model distillation (Hinton et al., 2014), where one trains a student model to imitate a teacher. There are several major differences between imitation and distillation. For distillation, the model architecture, training data, hyperparameters, etc., are known. In model imitation, one tries to imitate the teacher without this knowledge. Moreover, for distillation it is common to use training objectives that aim to match the output probability distribution of the teacher; in imitation this distribution may be unavailable.

LM Reverse Engineering and Re-implementation As companies deploy proprietary LMs such as ChatGPT and

Codex, numerous open-source and commercial efforts often quickly look to reproduce their capabilities. While not directly "stealing", these works instead look to probe the capabilities of the model and attempt to independently reimplement it. For example, organizations have announced efforts to reproduce Instruct-GPT (CarperAI, 2022), while other efforts have exposed specific model details such as the sampling parameters (Tay et al., 2020) or ChatGPT's hidden prompt (e.g., prompt injection attacks). Similarly, Thakkar (2023) expose the exact set of API calls and internal bookkeeping performed by GitHub CoPilot.

Data Memorization and Synthesis The key factor that enables model imitation is the ability of large LMs to generate high-quality synthetic data that is on par with human-annotated data. Indeed, past work argues that state-of-the-art generative models produce samples that are increasingly close to the true data distribution, and sometimes they even produce verbatim regenerations (Carlini et al., 2021b; Somepalli et al., 2022; Ho et al., 2020). Alarmingly, as models get larger they tend to regenerate verbatim data more often (Carlini et al., 2022; Jagielski et al., 2023) which suggests that broad model imitation may become easier in the future. At the same time, companies may look to continually improve their LMs (e.g., OpenAI improves their API). Fortunately, this creates a sort of cat-and-mouse game where adversaries will need to constantly recollect more imitation data in specific or new domains if they want to close the gap to their competitors.

Defending Against Model Stealing Many past works explore defenses against model stealing. One approach is to *detect* their occurrence, e.g., Juuti et al. (2019) detect adversaries who make out-of-distribution queries. It may be possible to detect model stealing for large LMs in a similar fashion because of its unique query distribution—one first generates instructions and then generates outputs for each instruction. Another line of work explores membership–inference-based defenses but amplifies their strength using *watermarks* (Zhang et al., 2018; Szyller et al., 2019; Krishna et al., 2020; Hisamoto et al., 2020). These are responses—sometimes incorrect or adversarial—that are designed to use as optimal probes to test if a model is a stolen copy of one's own. Finally, Orekondy et al. (2020) and Wallace et al. (2020) propose "prediction poisoning" defenses that shift the API's output distribution to hinder the learning of the imitation model.

6. Conclusion and Future Work

In this work, we proposed a framework for generating training data from proprietary LMs in order to imitate their performance on specific domains. We find that there is a strong correlation between the amount of data that we train on and the relative qualitative performance of the models on specific domains according to human evaluators.

We show that for specific domain distributions, this technique is relatively cheap and can yield qualitatively good results. In theory, it is possible to extend this method to fully imitate proprietary language models across all domains, but in order to do so, one would perhaps require unwieldy or costly amounts of domain specific data. We leave more thorough investigations of the limitations and failure modes of broad model imitation to future work.

Addressing Potential Ethical Concerns

Our goal is to make NLP models more secure against adversaries. To do so, we look to preempt possible harms and encourage more responsible model deployments. Nevertheless, releasing our paper does pose hypothetical real-world dangers. We take numerous steps to mitigate these harms.

First, we do not cause any damage to real-world users or companies. Our models do not compete with any company in any form and we will not make our imitation models available in any form to the public. We will also delete the models and datasets that we collected.

Second, although malicious actors could use our paper as inspiration for real attacks, there are still practical obstacles to them receiving financial benefit from them. For instance, if adversaries steal an API and try to launch an associated competitor service, they may face legal action depending on the terms of service for that API.

Finally, we note that two concurrent papers have already been posted online and use nearly identical methodology to our work, although with a different paper framing (Wang et al., 2022a; Honovich et al., 2022). Taken together, we believe that publishing our paper and publicly disclosing these vulnerabilities is thus both ethical and responsible.

Acknowledgements

We would like to thank Nicholas Carlini for valuable feedback during the course of the project. We would also like to thank Charlie Snell for providing the codebase we used to train these models.

References

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., et al. GPT-NeoX-20B: An open-source autoregressive language model. In *ACL Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan,

J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. Membership inference attacks from first principles. In *IEEE S&P*, 2021a.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., and Raffel, C. Extracting training data from large language models. In *USENIX Security Symposium*, 2021b.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

CarperAI. CarperAI, an EleutherAI lab, announces plans for the first open-source "instruction-tuned" language model. 2022. URL https://carper.ai/instruct-gpt-announcement/.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014.

Hisamoto, S., Post, M., and Duh, K. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system? In *TACL*, 2020.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Honovich, O., Scialom, T., Levy, O., and Schick, T. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*, 2022.

Jagielski, M., Thakkar, O., Tramér, F., Ippolito, D., Lee, K., Carlini, N., Wallace, E., Song, S., Thakurta, A., Papernot, N., and Zhang, C. Measuring forgetting of memorized training examples. In *ICLR*, 2023.

Juuti, M., Szyller, S., Marchal, S., and Asokan, N. PRADA: protecting against DNN model stealing attacks. In *IEEE EuroS&P*, 2019.

Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. Large language models struggle to learn long-tail knowledge. *arXiv preprint arXiv:2211.08411*, 2022.

Krishna, K., Tomar, G. S., Parikh, A. P., Papernot, N., and Iyyer, M. Thieves on sesame street! Model extraction of BERT-based APIs. In *ICLR*, 2020.

Lowd, D. and Meek, C. Adversarial learning. In *KDD*, 2005.

Metz, C. and Weise, K. Microsoft bets big on the creator of ChatGPT in race to dominate A.I. *New York Times*, 2023.

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. MetaICL: Learning to learn in context. In *NAACL*, 2022.

OpenAI. ChatGPT: Optimizing language models for dialogue. 2022. URL https://openai.com/blog/chatgpt/.

Orekondy, T., Schiele, B., and Fritz, M. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, 2019.

Orekondy, T., Schiele, B., and Fritz, M. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *ICLR*, 2020.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Pal, S., Gupta, Y., Shukla, A., Kanade, A., Shevade, S., and Ganapathy, V. A framework for the extraction of deep neural networks by leveraging public data. *arXiv preprint arXiv:1905.09165*, 2019.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*, 2022.

Szyller, S., Atli, B. G., Marchal, S., and Asokan, N. DAWN: Dynamic adversarial watermarking of neural networks. In *ACM Multimedia*, 2019.

Tay, Y., Bahri, D., Zheng, C., Brunk, C., Metzler, D., and Tomkins, A. Reverse engineering configurations of neural text generation models. In *ACL*, 2020.

Thakkar, P. CoPilot internals, 2023. URL https://thakkarparth007.github.io/copilot-explorer/posts/copilot-internals.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, 2016.

Wallace, E., Stern, M., and Song, D. Imitation attacks and defenses for black-box machine translation systems. In *EMNLP*, 2020.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-Instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022a.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Arunkumar, A., Ashok, A., Dhanasekaran, A. S., Naik, A., Stap, D., et al. Benchmarking generalization via in-context instructions on 1,600+ language tasks. In *EMNLP*, 2022b.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., and Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In *ACM ASIACCS*, 2018.

Zhong, R., Lee, K., Zhang, Z., and Klein, D. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *EMNLP*, 2021.

A. Task Categories

We focus on imitating the task categories from Ouyang et al. (2022). We include details on each category:

- **Text generation** tasks may construct some hypothetical scenario and ask for an opinion or what one would do in the scenario. They are generally open-ended based on the context of the prompt.
- **Brainstorming** tasks typically ask for opinions or ideas on how to do something.
- **Chat** tasks will construct some scenario, and the response will be formatted as a chat dialogue between one or more people.
- For **question answering**, there are two main types, those that can have any answer (openQA), and those that have an answer from a fixed set of answer choices or context (closedQA)
- **Summarization** tasks will ask provide instructions on how to summarize a particular text input.
- **Rewriting** and **extraction** tasks will provide instructions on how to rewrite or extract artifacts from text respectively.
- **Classification** tasks provide instructions on how to classify an input, usually providing the label set.

For concrete examples of these task instructions, refer to the seed tasks in Table 3.

B. Seed Task List

In Table 3, we present the handwritten tasks that were used to seed our task instruction generation pipeline. These task instructions were chosen since they are illustrative of Instruct-GPTs capabilities on various broad-coverage categories. For each task category, we write 10 task instructions in a list format and then ask Instruct-GPT to continue filling out the list while focusing on being creative and descriptive. Each time the model is queried, 3 of the seed tasks are randomly chosen to elicit a new task instruction in a particular category. Once the task pool exceeds 20 in each category, we sample 3 instructions from the pool of seed and previously-generated instruction to elicit a new task. In Table 4, we detail the prompt template that is used to generate new tasks.

For generating task instructions, we query text-davinci-002 with temperature=0.8 and max_tokens=512. All of the generations that match the list format specified in Table 4 are then compared for similarity to existing task instructions in the pool. The instructions which have a maximum BLEU score similarity of under 20% to any previously generated instruction are accepted into the pool.

B.1. Prompt Templates

We use two-main prompting templates to synthesize either output-only or input-only instances from Instruct-GPT. For output-only tasks such as brainstorming, generation and openQA, we use the template in Table 5. This is the equivalent of simply pasting the generated task instruction back into the model. For categories such as extraction, rewriting, summarization, classification and closedQA, we use the template in Table 6, which asks the model to generate both an input and output pair. For more structured categories like chat, we make a small modification and use the template in Table 7, which asks the model to format the output in the style of a chat dialogue. Additionally, for closedQA, which requires a set of answer choices to pick from, we use the prompting template in Table 8. We note that these templates are generally straightforward and easy to modify to collect synthetic data from different categories.

For generating instance examples, we query text-davinci-002 with temperature=0.7 and max_tokens=1024. If the length of the output is sufficiently large (over 10 tokens), we store the generation in the dataset.

B.2. Amazon Mechanical Turk Interface

For benchmarking our imitation models against Instruct-GPT, we use Amazon Mechanical Turk with the UI in Figure 5. It shows human evaluators a random task instruction and output response from one of our models. Evaluators then rate the quality of the output from 1–5 using various criteria. For our human evaluations, we rate outputs on each of the 500 instructions in the evaluation set. We report overall ratings as well as a per-category breakdown.

To evaluate input-output tasks, we also ask human annotators to create inputs to use. We present the UI for this collection task in Figure 6. We provide instructions and samples in the sidebar. We collect approximately 85 inputs for input-output task instructions in the evaluation set.

For our ratings evaluations, we collect 3 unique ratings for every example in the evaluation set. For all evaluations, roughly 100 unique human evaluators were used. We pay these evaluators roughly $15/hour based on the average time it takes to complete a task. In total, we spend roughly $5000 on our ratings experiments and $500 on collecting inputs, inclusive of service fees.

C. SuperNI Results

Table 9 shows the results on SuperNI.

Task Seed List

Brainstorming

- Think of new ways to make nuclear energy more financially reasonable
- Brainstorm different ideas to solve traffic congestion in Los Angeles
- There is a school fundraiser coming up, do you have any ideas?
- Help a student get better at his studies
- List the pros and cons of implementing a carbon tax
- How can I improve my productivity?
- Come up with methods for making your community more sustainable
- Generate ideas for reducing your carbon footprint
- What are some fun and unique ways to exercise?
- Ideas for a fun weekend activity

Chat

- Marv is a chatbot that reluctantly answers questions with sarcastic responses.
- This is a conversation with an enlightened Buddha. Every response is full of wisdom and love.
- The following is a conversation with an AI assistant. The assistant is helpful, creative, clever, and very friendly.
- Bob is a 1st year PhD student at UC Berkeley and Lacy is his advisor.
- This is a conversation with Steven. Steven likes to watch Netflix and hasn't left his home in 2 weeks.
- The following is a conversation between you and your friend group. Plan a hiking trip for the weekend.
- A new mother and father are talking about baby names.
- Ask a friend or family member for recommendations on something you're interested in (e.g., books, movies, music, etc.).
- You're standing in line at a coffee shop. Make small talk with the barista.
- John and Mike are coworkers who are complaining to each other about their job.

Classification

- Given a short description of a made-up animal, classify it as either a bird or a mammal.
- Take a text document and determine if it is about cats or dogs.
- For each piece of text, classify it as positive, negative, or neutral.
- Given a list of movie reviews, label each review as positive or negative.
- Classify the following texts as either fiction or non-fiction.
- Classify the following news headlines as being about domestic or international news.
- Given a list of Amazon reviews, classify each review as positive or negative.
- Determine if a given text document is about the topic of 'politics'
- Determine whether a given text document is about the topic of baseball or the topic of basketball.
- Given a piece of text, classify it as belonging to one of three genres: mystery, romance, or comedy.

ClosedQA

- What happens when you drop a heavy stone from a tree?
- Choose the ingredients that are needed to make a meatball marinara.
- Which number is closest to pi?
- Which item would you take on a tropical trip?
- What is the best way to study for a test?
- What are the medical symptoms of the flu?
- How many different kinds of animals are there in the world?
- What is the capital of Australia?
- How would you make a simple meatball marinara dish?
- What are the lyrics to the chorus of the song 'Satisfied'

Extract

- Given the following list of movie titles, write down any names of cities in the titles.
- Given this table, find all the rows where the value in the second column is greater than the value in the first column.
- Find all the words that are longer than 5 letters in this text.
- Find all the instances of the author using first-person point of view in this blog post.
- Extract all the course titles from the following course list.

- Given this list of words, find all the palindromes.
- Given this list of numbers, find all the prime numbers.
- Find the average of all the numbers in this list.
- Find all the phone numbers in this document.
- Find the line of best fit for this data.

Generation

- Create a made-up creature and describe its habitat, diet, and appearance
- Write a short story from the perspective of an inanimate object
- Have a conversation between two people, with each piece of dialogue being one word longer than the last
- Write a children's story that only uses words that start with the letter 't'
- Create a character and write their life story from birth to death
- Invent a new holiday and describe how it is celebrated
- Create a map of an island that you make up
- Write a poem about your favorite season
- Create a menu for a 5-star restaurant
- You're a alien on a new planet. What do you do?

OpenQA

- Who built the leaning tower of Pisa?
- Where did the word "kindergarten" originate from?
- Who wrote the novel 'Pride and Prejudice'?
- What are the Ten Commandments?
- What is the definition of the word "pejorative"?
- How many calories are in a banana?
- What is the square root of pi?
- How do you do a breadth first search?
- How do you take a derivative of a function?
- What is the 6th element on the periodic table of elements?

Rewrite

- Edit this paragraph for spelling mistakes and clarity.
- Remove all punctuation from this text.
- Remove all the duplicates from this list.
- Rewrite this email to make it sound more professional.
- Translate this document from French to English.
- Make this product description more persuasive.
- Rewrite this short story to be more suspenseful.
- Take this research paper and make it more accessible to a lay audience.
- Edit the rhyme scheme of this poem to be more consistent.
- Rewrite this job ad to attract more qualified candidates.

Summarization

- Given a text passage, identify the main ideas and produce a summary of the text.
- Summarize this text for a second-grade student.
- Summarize the following conversation between a customer and customer assistant. Make sure to state any complaints of the customer.
- Determine the main idea of the text.
- Create a one-sentence summary of the plot of this book.
- Tell me the main points from that book you just read.
- Give me a overview of what this research paper is about.
- Summarize the gist of this article for me.
- Give me the cliff notes for the following book.
- Provide a brief overview of what happened in this novel.

Table 3: The handwritten task instructions that were used to seed the task instruction generation pipeline. These task instructions were chosen since they are illustrative of Instruct-GPT's capabilities on various broad-coverage categories.

List examples of {category} tasks that you can do. Focus on being as creative and descriptive as possible. Answer in the following format '- <>'. Replace <> with your answer. - {instruction for seed or sampled task 1} - {instruction for seed or sampled task 2} - {instruction for seed or sampled task 3} -

Table 4. Prompting template used for generating new instructions in any of the categories.

{task instruction from the output-only category}.

Table 5. Prompting template used to generate output-only examples for the categories of brainstorming, generation, and open-ended question answering (openQA).

This is a task instruction: {task instruction from an input-output category}. Read the task instruction and write an output for the following input:

Table 6. Prompting template used to generate input and output pairs for text extraction, rewriting, summarization and classification.

This is a task instruction: {task instruction from the chat category}. Give me an example for this task instruction that is formatted like a chat conversation between people. Focus on making the format as varied as possible.

Table 7. Prompting template used to generate formatted dialogue for the chat category.

This is a question: {task instruction from closedQA category}. Read the question and pick the best answer among the following options:

Table 8. Prompting template used to generate answer choices for closed-ended question and answering tasks (closedQA).

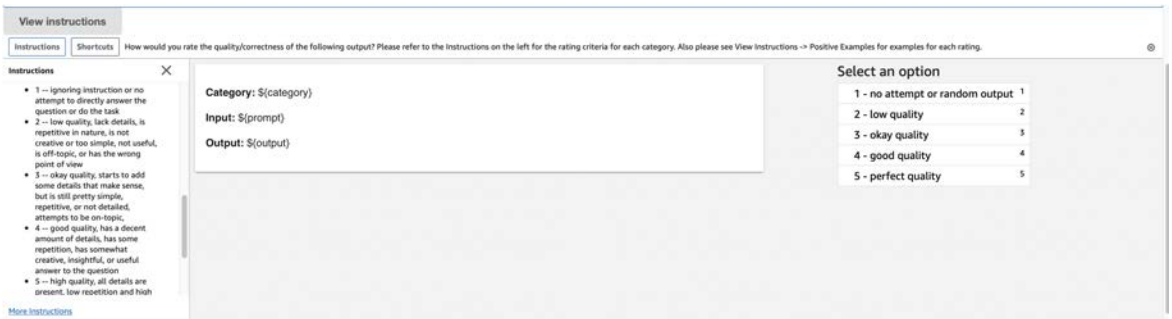


Figure 5. Our Amazon Mechanical Turk interface for measuring the quality and correctness of model outputs for instruction following tasks. Evaluators are presented with the category, instruction, and output for a random model and are told to rate the output from 1–5 using various criteria.

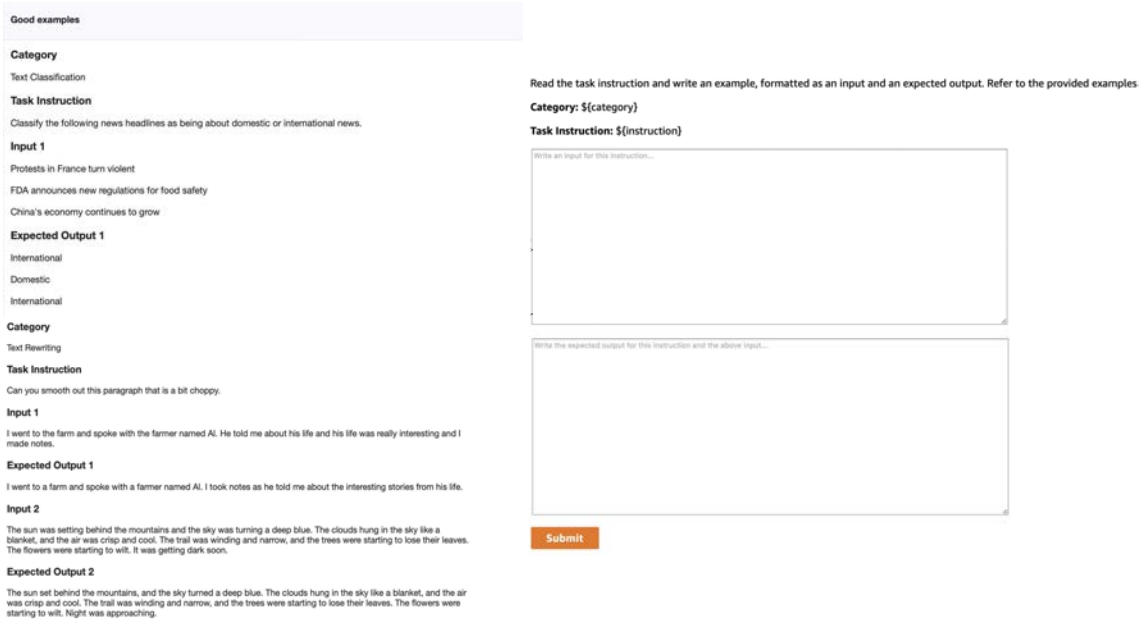


Figure 6. Our Amazon Mechanical Turk interface for collecting inputs for input-output tasks in the holdout evaluation set. We provide some samples and ask annotators to write an input and output pair for each task instruction in the holdout set. We save the corresponding human-written inputs to be used during evaluation.

		Imitation Models				
Category	**GPT-J**	**10K**	**50K**	**100K**	**250K**	**Instruct-GPT**
Textual Entailment	2	15	24	13	6	41
Cause Effect Classification	4	18	28	14	10	38
Coreference Resolution	2	15	19	13	11	31
Dialogue Act Recognition	2	9	15	10	6	63
Answerability Classification	1	12	18	12	6	52
Word Analogy	1	6	8	5	4	27
Overlap Extraction	3	20	17	16	15	27
Keyword Tagging	1	13	14	10	8	20
Question Rewriting	6	34	29	29	32	48
Title Generation	2	13	14	13	12	32
Data To Text	7	26	24	22	21	28
Grammar Error Correction	10	75	61	53	54	52
Average	3	16	**20**	15	12	38

Table 9. Automated evaluations. We evaluate our imitation models on zero-shot SuperNI (Wang et al., 2022b) and report the average Rouge-L score between the model and reference outputs for each category. Our imitation models can close approximately half of the gap from the pre-trained GPT-J model to Instruct-GPT. We note that SuperNI comes from a different distribution than the data that we query to the API. In fact, the Instruct-GPT paper notes that as they collect and fine-tune on larger internal datasets, their model actually gets worse on public benchmarks. Therefore, we do not expect our models to perform particularly well on this evaluation.

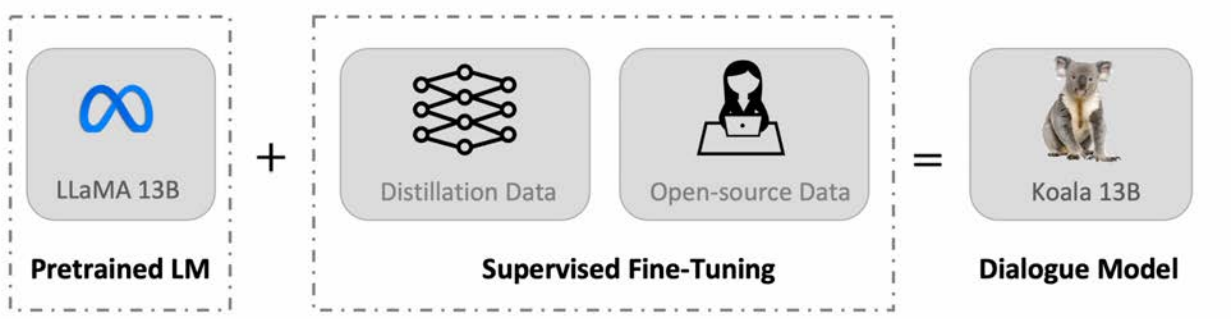
Koala: A Dialogue Model for Academic Research

In this post, we introduce Koala, a chatbot trained by fine-tuning Meta's LLaMA on dialogue data gathered from the web. We describe the dataset curation and training process of our model, and also present the results of a user study that compares our model to ChatGPT and Stanford's Alpaca. Our results show that Koala can effectively respond to a variety of user queries, generating responses that are often preferred over Alpaca, and at least tied with ChatGPT in over half of the cases.

We hope that these results contribute further to the discourse around the relative performance of large closed-source models to smaller public models. In particular, it suggests that models that are small enough to be run locally can capture much of the performance of their larger cousins if trained on carefully sourced data. This might imply, for example, that the community should put more effort into curating high-quality datasets, as this might do more to enable safer, more factual, and more capable models than simply increasing the size of existing systems. We emphasize that Koala is a research prototype, and while we hope that its release will provide a valuable community resource, it still has major shortcomings in terms of content, safety, and reliability, and should not be used outside of research.

Online interactive demo Training and serving framework on GitHub

System Overview



Large language models (LLMs) have enabled increasingly powerful virtual assistants and chat bots, with systems such as ChatGPT, Bard, Bing Chat, and Claude able to respond to a breadth of user queries, provide sample code, and even write poetry. Many of the most capable LLMs require huge computational resources to train, and oftentimes use large and proprietary datasets. This suggests that in the future, highly capable LLMs will be largely controlled by a small number of organizations, and both users and researchers will pay to interact with these models without direct access to modify and improve them on their own. On the other hand, recent months have also seen the release of increasingly capable freely available or (partially)

open-source models, such as LLaMA. These systems typically fall short of the most capable closed models, but their capabilities have been rapidly improving. This presents the community with an important question: will the future see increasingly more consolidation around a handful of closed-source models, or the growth of open models with smaller architectures that approach the performance of their larger but closed-source cousins?

While the open models are unlikely to match the scale of closed-source models, perhaps the use of carefully selected training data can enable them to approach their performance. In fact, efforts such as Stanford's Alpaca, which fine-tunes LLaMA on data from OpenAI's GPT model, suggest that the right data can improve smaller open source models significantly.

We introduce a new model, Koala, which provides an additional piece of evidence toward this discussion. Koala is fine-tuned on freely available interaction data scraped from the web, but with a specific focus on data that includes interaction with highly capable closed-source models such as ChatGPT. We fine-tune a LLaMA base model on dialogue data scraped from the web and public datasets, which includes high-quality responses to user queries from other large language models, as well as question answering datasets and human feedback datasets. The resulting model, Koala-13B, shows competitive performance to existing models as suggested by our human evaluation on real-world user prompts.

Our results suggest that learning from high-quality datasets can mitigate some of the shortcomings of smaller models, maybe even matching the capabilities of large closed-source models in the future. This might imply, for example, that the community should put more effort into curating high-quality datasets, as this might do more to enable safer, more factual, and more capable models than simply increasing the size of existing systems.

By encouraging researchers to engage with our system demo, we hope to uncover any unexpected features or deficiencies that will help us evaluate the models in the future. We ask researchers to report any alarming actions they observe in our web demo to help us comprehend and address any issues. As with any release, there are risks, and we will detail our reasoning for this public release later in this blog post. We emphasize that Koala is a research prototype, and while we hope that its release will provide a valuable community resource, it still has major shortcomings in terms of content, safety, and reliability, and should not be used outside of research. Below we provide an overview of the differences between Koala and notable existing models.

Model	Training Set	Training Code	Public Weights	Dialogue Fine-tuned	Evaluation Method
Alpaca	OpenAI API outputs	✔	✔	✗	evaluation by 5 humans
ChatGPT	proprietary	✗	✗	✔	proprietary
Koala	Public dialogues & preferences	✔	✔	✔	evaluation by 100 humans

Datasets and Training

A primary obstacle in building dialogue models is curating training data. Prominent chat models, including ChatGPT, Bard, and Bing Chat, use proprietary datasets built using significant amounts of human annotation. To construct Koala, we curated our training set by gathering dialogue data from the web and public datasets. Part of this data includes dialogues with large language models (e.g., ChatGPT) which users have posted online.

Rather than maximizing *quantity* by scraping as much web data as possible, we focus on collecting a small *high-quality* dataset. We use public datasets for question answering, human feedback (responses rated both positively and negatively), and dialogues with existing language models. We provide the specific details of the dataset composition below.

ChatGPT Distillation Data

Public User-Shared Dialogues with ChatGPT (ShareGPT) Around 60K dialogues shared by users on ShareGPT were collected using public APIs. To maintain data quality, we deduplicated on the user-query level and removed any non-English conversations. This leaves approximately 30K examples.

Human ChatGPT Comparison Corpus (HC3) We use both the human and ChatGPT responses from the HC3 english dataset, which contains around 60K human answers and 27K ChatGPT answers for around 24K questions, resulting in a total number of around 87K question-answer examples.

Open Source Data

Open Instruction Generalist (OIG)
We use a manually-selected subset of components from the Open Instruction Generalist dataset curated by LAION. Specifically, we use the grade-school-math-instructions, the poetry-to-songs, and the plot-screenplay-books-dialogue datasets. This results in a total of around 30k examples.

Stanford Alpaca
We include the dataset used to train the Stanford Alpaca model. The dataset contains around 52K examples, which is generated by OpenAI's text-davinci-003 following the self-instruct process. It is worth noting that HC3, OIG, and Alpaca datasets are single-turn question answering while ShareGPT dataset is dialogue conversations.

Anthropic HH
The Anthropic HH dataset contains human ratings of harmfulness and helpfulness of model outputs. The dataset contains ~160K human-rated examples, where each example in this dataset consists of a pair of responses from a chatbot, one of which is preferred by humans. This dataset provides both capabilities and additional safety protections for our model.

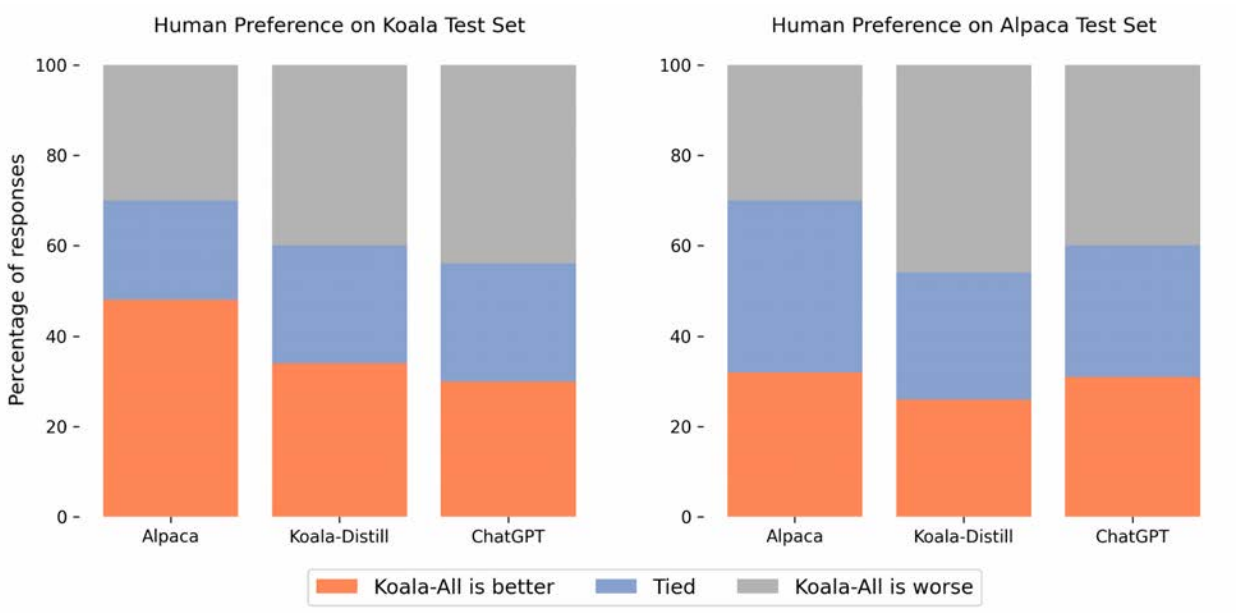
OpenAI WebGPT
The OpenAI WebGPT dataset includes a total of around 20K comparisons where each example comprises a question, a pair of model answers, and metadata. The answers are rated by humans with a preference score.

OpenAI Summarization
The OpenAI summarization dataset contains ~93K examples, each example consists of feedback from humans regarding the summarizations generated by a model. Human evaluators chose the superior summary from two options.

When using the open-source datasets, some of the datasets have two responses, corresponding to responses rated as good or bad (Anthropic HH, WebGPT, OpenAI Summarization). We build on prior research by Keskar et al, Liu et al, and Korbak et al, who demonstrate the effectiveness of conditioning language models on human preference markers (such as “a helpful answer” and “an unhelpful answer”) for improved performance. We condition the model on either a positive or negative marker depending on the preference label. We use positive markers for the datasets without human feedback. For evaluation, we prompt models with positive markers.

The Koala model is implemented with JAX/Flax in EasyLM, our open source framework that makes it easy to pre-train, fine-tune, serve, and evaluate various large language models. We train our Koala model on a single Nvidia DGX server with 8 A100 GPUs. It takes 6 hours to complete the training for 2 epochs. On public cloud computing platforms, such a training run typically costs less than $100 with preemptible instances.

Preliminary Evaluation



In our experiments, we evaluated two models: Koala-Distill, which solely employs distillation data, and Koala-All, which employs all of the data, including both distillation and open-source data. Our aim is to compare the performance of these models and evaluate the influence of distillation and open-source datasets on final performance. We ran a human evaluation to compare Koala-All with Koala-Distill, Alpaca, and ChatGPT. We present our results in the figure above. We evaluate on two different sets, one consisting of 180 test queries used by Stanford's Alpaca ("Alpaca Test Set"), and our own test set ("Koala Test Set").

The Alpaca test set consists of user prompts sampled from the self-instruct dataset, and represents in-distribution data for the Alpaca model. To provide a second more realistic evaluation protocol, we also introduce our own (Koala) test set, which consists of 180 *real* user queries that were posted online. These user queries span various topics, are generally conversational in style, and are likely more representative of the real-world use cases of chat-based systems. To mitigate possible test-set leakage, we filtered out queries that have a BLEU score greater than 20% with any example from our training set. Additionally, we removed non-English and coding-related prompts, since responses to these queries cannot be reliably reviewed by our pool of raters (crowd workers). We release our test-set for academic use and future benchmarking.

With these two evaluation sets, we conducted a blind pairwise comparison by asking approximately 100 evaluators on Amazon Mechanical Turk platform to compare the quality of model outputs on these held-out sets of prompts. In the ratings interface, we present each rater with an input prompt and the output of two models. They are then asked to judge which output is better (or that they are equally good) using criteria related to response quality and correctness.

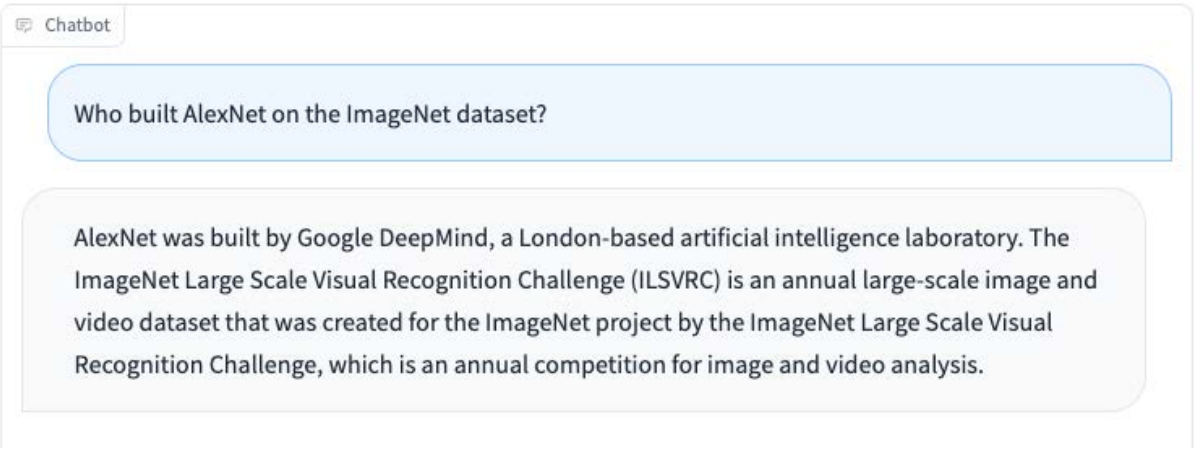
On the Alpaca test set, Koala-All exhibited comparable performance to Alpaca. However, on our proposed test set, which consists of real user queries, Koala-All was rated as better than Alpaca

in nearly half the cases, and either exceeded or tied Alpaca in 70% of the cases. Of course, the more conversational prompts in the Koala test set more closely resemble the Koala training set, so this is perhaps not surprising, but insofar as such prompts more closely resemble likely downstream use cases for such models, this suggests that Koala would be expected to perform better in assistant-like applications. This suggests that data of LLM interactions sourced from examples posted by users on the web is an effective strategy for endowing such models with effective instruction execution capabilities.

Perhaps more surprisingly, we found that training on open-source data in addition to the distillation data (Koala-All) performs slightly worse than training on just ChatGPT distillation data (Koala-Distill), as shown by the comparison to Koala-Distill on both datasets. Though the difference might not be significant, this result suggests that the ChatGPT dialogues are of such high quality that incorporating even twice as much open-source data did not lead to a significant improvement. Our initial hypothesis was that Koala-All should perform at least somewhat better, hence we used it as our primary model in all evaluations, but a potential takeaway from these experiments is that effective instruction and assistant models could be finetuned from LLM backbones such as LLaMA entirely using data from larger and more powerful models, so long as the prompts for these responses are representative of the kinds of prompts that users will provide at test-time. This also further supports the notion that the key to building strong dialogue models may lie more in curating high-quality dialogue data that is diverse in user queries, rather than simply reformatting existing datasets as questions and answers.

Limitations and Safety

Like other language models, Koala has limitations and can be harmful when misused. We observe that Koala can hallucinate and generate non-factual responses with a highly confident tone, which is likely a result of the dialogue fine-tuning. Perhaps an unfortunate implication of this is that smaller models inherit the confident *style* of larger language models before they inherit the same level of factuality—if true, this is a limitation that is important to study in future work. When misused, the hallucinated responses from Koala can potentially facilitate the spread of misinformation, spam, and other content.



Koalas can hallucinate inaccurate information in a confident and convincing tone.

Koalas can hallucinate inaccurate information in a confident and convincing tone. Beyond hallucinations, Koala shares deficiencies from other chatbot language models. Some of which include:

- Biases and Stereotypes: Our model will inherit biases from the dialogue data it was trained on, possibly perpetuating harmful stereotypes, discrimination, and other harms.
- Lack of Common Sense: While large language models can generate text that appears to be coherent and grammatically correct, they often lack common sense knowledge that humans take for granted. This can lead to nonsensical or inappropriate responses.
- Limited Understanding: Large language models can struggle to understand the context and nuances of a dialogue. They can also have difficulty identifying sarcasm or irony, which can lead to misunderstandings.

To address the safety implications of Koala, we included adversarial prompts in the dataset from ShareGPT and Anthropic HH to make the model more robust and harmless. To further mitigate potential misuse, we deploy OpenAI's content moderation filter in our online demo to flag and remove unsafe content. We will be cautious about the safety of Koala, and we are committed to perform further safety evaluations of it while also monitoring our interactive demo. Overall, we decided to release Koala because we think its benefits outweigh its risks.

Release

We are releasing the following artifacts

- An online interactive demo of Koala
- EasyLM: our open source framework we used to train Koala
- The code for preprocessing our training data
- Our test set of queries

- Weights: we hope to release the weights soon, but we are still investigating some permissions and licensing questions to do this responsibly.

License

The online demo is a research preview intended for academic research only, subject to the model License of LLaMA, Terms of Use of the data generated by OpenAI, and Privacy Practices of ShareGPT. Any other usage of the online demo, including but not limited to commercial usage, is strictly prohibited. Please contact us If you find any potential violations. Our training and inference code is released under the Apache License 2.0.

Future Work

We hope that the Koala model will serve as a useful platform for future academic research on large language models: the model is capable enough to exhibit many of the capabilities that we associate with modern LLMs, while being small enough to be finetuned or utilized with more limited compute. Potentially promising directions might include:

- Safety and alignment: Koala allows further study of language model safety and better alignment with human intentions.
- Model bias: Koala enables us to better understand the biases of large language models, the presence of spurious correlations and quality issues in dialogue datasets, and methods to mitigate such biases.
- Understanding large language models: because Koala inference can be performed on relatively inexpensive commodity GPUs, it enables us to better inspect and understand the internals of dialogue language models, making (previously black-box) language models more interpretable.

The Team

The Koala model is a joint effort across multiple research groups in the Berkeley Artificial Intelligence Research Lab (BAIR) of UC Berkeley.

Students (alphabetical order):
Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace

Advisors (alphabetical order):
Pieter Abbeel, Sergey Levine, Dawn Song

Acknowledgments

We express our gratitude to Sky Computing Lab at UC Berkeley for providing us with serving backend support. We would like to thank Charlie Snell, Lianmin Zheng, Zhuohan Li, Hao Zhang, Wei-Lin Chiang, Zhanghao Wu, Aviral Kumar and Marwa Abdulhai for discussion and feedback. We would like to thank Tatsunori Hashimoto and Jacob Steinhardt for discussion around limitations and safety. We would also like to thank Yuqing Du and Ritwik Gupta for helping with the BAIR blog. Please check out the blog post from Sky Computing Lab about a concurrent effort on their chatbot, Vicuna.