Machine Intelligence Research
www.mi-research.net
DOI: 10.1007/s11633-024-1502-8

MOSS: An Open Conversational Large Language Model

Tianxiang Sun Xiaotian Zhang Zhengfu He Peng Li Qinyuan Cheng
Xiangyang Liu Hang Yan Yunfan Shao Qiong Tang Shiduo Zhang
Xingjian Zhao Ke Chen Yining Zheng Zhejian Zhou Ruixiao Li
Jun Zhan Yunhua Zhou Linyang Li Xiaogui Yang Lingling Wu
Zhangyue Yin Xuanjing Huang Yu-Gang Jiang Xipeng Qiu

Fudan University, Shanghai 200438, China

Abstract: Conversational large language models (LLMs) such as ChatGPT and GPT-4 have recently exhibited remarkable capabilities across various domains, capturing widespread attention from the public. To facilitate this line of research, in this paper, we report the development of MOSS, an open-sourced conversational LLM that contains 16 B parameters and can perform a variety of instructions in multi-turn interactions with humans. The base model of MOSS is pre-trained on large-scale unlabeled English, Chinese, and code data. To optimize the model for dialogue, we generate 1.1 M synthetic conversations based on user prompts collected through our earlier versions of the model API. We then perform preference-aware training on preference data annotated from AI feedback. Evaluation results on real-world use cases and academic benchmarks demonstrate the effectiveness of the proposed approaches. In addition, we present an effective practice to augment MOSS with several external tools. Through the development of MOSS, we have established a complete technical roadmap for large language models from pre-training, supervised fine-tuning to alignment, verifying the feasibility of chatGPT under resource-limited conditions and providing a reference for both the academic and industrial communities. Model weights and code are publicly available at https://github.com/OpenMOSS/MOSS.

Keywords: Large language models, natural language processing, pre-training, alignment, chatGPT, MOSS.

Citation: T. Sun, X. Zhang, Z. He, P. Li, Q. Cheng, X. Liu, H. Yan, Y. Shao, Q. Tang, S. Zhang, X. Zhao, K. Chen, Y. Zheng, Z. Zhou, R. Li, J. Zhan, Y. Zhou, L. Li, X. Yang, L. Wu, Z. Yin, X. Huang, Y. G. Jiang, X. Qiu. MOSS: An open conversational large language model. *Machine Intelligence Research*. http://doi.org/10.1007/s11633-024-1502-8

1 Introduction

Large language models (LLMs)[1] have demonstrated unprecedented capabilities on a variety of language tasks, such as GPT-3[2], Gopher[3], PaLM[4], Chinchilla[5], GLM-130B[6], LLaMA[7], and GPT-4[8]. After aligning with human preferences, these LLMs can serve as capable AI assistants that are helpful across many domains[8–11]. Such AI assistants are usually trained to interact with users in a conversational manner, capturing widespread attention from not only the research community but also the public.

The unprecedented intelligence that LLMs exhibit goes beyond some of the authors might expect a probabilistic model to have. One speculative explanation to this is that the model learns some concepts abstract and general enough to function in more situations than it has been trained on. The impetus of LLMs to learn such complicated structure of world knowledge by merely imitating, i.e., minimizing the loss of the next token might lie in that language itself implies human cognition of the world′s logic. And with the size of the model scales, the model learns more general concepts and is able to handle more situations i.e. a better compression of knowledge[12]. The idea of training MOSS serves as a first step to validate and, if this is true to an acceptable extent, implement a Chinese version of prototype of this promising path.

Despite the success and popularity of large-scale AI assistants, at the start time of this work, few studies have been publicly disclosed due to the expensive annotation and training costs. To that end, we present MOSS, an open-sourced conversational LLM with 16 B parameters. As depicted in Fig. 1, the development of MOSS includes three stages: cross-lingual pre-training, supervised fine-tuning, and preference-aware training. Compared with existing efforts (e.g., LLaMA[7] and Stanford Alpaca[1]) in the open-source community, MOSS is featured by:

1. **Cross-lingual Pre-training.** At the inception of

Research Article
Manuscript received on December 25, 2023; accepted on March 15, 2024
Recommended by Associate Editor Zhiyuan Liu
Colored figures are available in the online version at https://link.springer.com/journal/11633

[1] In fact, the launch time of this work is much earlier than LLaMA and Stanford Alpaca, so there are few publicly available models for comparison.

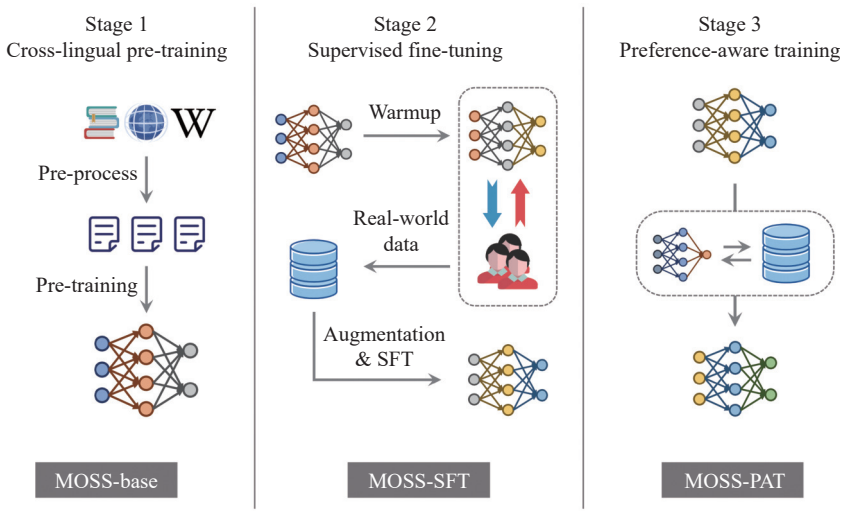


Fig. 1 The development of MOSS includes three stages. In stage 1 (Section 3), we pre-train the cross-lingual MOSS-base model with public text and code corpora. In stage 2 (Sections 5 and 6), we first perform supervised fine-tuning (SFT) with synthetic conversational data and deploy it to the public. We then use the collected real-world data as a seed set to synthesize a new training set, which is used to perform the final SFT. In stage 3 (Section 7), we train a preference model and use it to perform preference-aware training. The models resulting from the three stages are named MOSS-base, MOSS-SFT, and MOSS-PAT, respectively.

the MOSS project, we encountered significant challenges in training a large-scale, purely Chinese model (such as CPT[13] or Chinese BART[13]) to function as a versatile AI assistant. To address this, we initiated the pre-training of the MOSS base model on a diverse dataset comprising 360 B English tokens (predominantly sourced from the Pile[14]), 100 B Chinese tokens (largely derived from proprietary datasets), and 220 B code tokens (mainly extracted from the Pile, BigQuery, and BigPython). This strategy was instrumental in validating our hypothesis that knowledge transfer between Chinese and English is feasible, even in the absence of direct sentence-level alignment between the two languages.

2. **Helpful, honest, and harmless (HHH).** In contrast to most existing open-sourced models that mainly focus on improving helpfulness, MOSS is also designed to be honest[15] and harmless. We collect and extend honesty- and harmlessness-relevant conversational data for supervised fine-tuning (SFT). In addition, we perform preference-aware training on additional data to ensure MOSS is aware of the quality of its response in helpfulness.

3. **Alignment with the real-world distribution of user intents.** Real-world user prompts are inevitably diverse, making it difficult to optimize LLMs targeting user intents. To this end, we deployed an early version of MOSS and collected 100 K user prompts submitted through the web application. Our SFT data and preference data are synthesized from a filtered subset of the user prompts, ensuring that the training data of MOSS and the real-world user intents are identically distributed.

4. **Preference-aware training.** Aligning LLMs with human preferences is becoming a necessary step before the public release[8], which significantly improves model usability and harmlessness. Existing alignment research usually requires a preference model (also referred to as reward model) trained from human feedback[9] or AI feedback[16] to measure the quality of model responses to human preferences. The preference model can be used for performing rejection sampling[17] or reinforcement learning[9, 10]. The former approach is inefficient as it requires the model to generate multiple responses at inference time. The latter one, a.k.a. reinforcement learning from human feedback (RLHF), is sensitive to hyperparameters and therefore is practically hard to tune. Instead, we employ a preference model to tag model responses with their overall quality. These tags are prepended to the model responses for each round of the conversation. By performing conventional fine-tuning on such preference-tagged conversational data, MOSS is capable of distinguishing high-quality responses from low-quality ones. At inference time, MOSS can generate desired responses conditioning on specific preference tags, for instance, <quality:100>.

5. **Augmentation with tools.** Probabilistic language models are notorious for suffering from "hallucinations", e.g., they often generate outputs containing factual errors or basic arithmetic mistakes. Inspired by recent work in tool-augmented LLMs[18, 19], we perform a tool-oriented training to augment MOSS with several tools, i.e., search engine, calculator, equation solver, and text-to-image generator. Though the capability of the model is not fundamentally improved, we observe significant benefits when allowing MOSS to access external tools to answer user queries.

We conduct automatic evaluations for MOSS, demonstrating significant improvement over its base model and concurrent chat models in terms of model capabilities and

real-world user experiences.

2 Large language models

Transformer-based auto-regressive language models[20, 21], such as GPT-3[2], PaLM[4], Chinchilla[5], GLM-130B[6], LLaMA[7, 22], and GPT-4[8] have shown increasing power in solving diverse real-world tasks. At its core, the model performs next token prediction, which can instantiate most of the language-related tasks. Thus, achieving higher accuracy on next token prediction usually implies higher performance on many downstream language processing tasks. Fortunately, it has been shown that increasing the number of training tokens and model parameters can predictably improve the accuracy of next-token prediction[5, 12]. Since then, scaling up language models has become a promising way to overcome more and more tasks and unlocks more emergent abilities which smaller models do not exhibit[23].

Based on the pre-trained language model, ChatGPT performs alignment with human preferences and achieves remarkable performance when interacting with humans. The alignment consists of three steps: 1) Supervised fine-tuning (SFT), which is to fine-tune the language model to follow user instructions and the conversation format. 2) Reward modeling (RM), which is to collect a set of human-annotated preference data and train a reward model to mimic human preferences. 3) Reinforcement learning from human feedback (RLHF), which is to train a policy model with RL algorithms against the reward model. Such a training pipeline has been shown effective in the field of text summarization[24], single-turn instruction following[9], and multi-turn dialogue[8]. Despite its success, the high annotation cost (collecting human-written SFT data and preference data) and optimization difficulty (the PPO[25] algorithm used in RLHF) hinder its applications. Thus, much effort has been devoted to synthetic data and AI feedback. As a representative, AnthropicLM[16, 26] explored prompting language models to synthesize instruction and feedback data. Our work also lies in this line of research.

Fig. 2 presents a brief overview of LLMs published within 8 months after the release of ChatGPT. MOSS is one of the pioneer conversational LLMs. Table 1 provides a detailed comparison between MOSS and other concurrent open-sourced LLMs.

In comparison, LLMs released after MOSS were usually pre-trained using much more tokens and more advanced architectures (e.g., the LLaMA Transformer architecture[7, 22]). Though, MOSS achieved competitive interactive experiences by employing strong language models to generate complicated training signals, including multi-turn conversational data, preference data, and tool-augmented data. Without any human annotation and RL algorithms, MOSS achieved good performance as an AI assistant, providing practice for building conversational language models in a cheaper and faster fashion.

3 Cross-lingual pre-training

Building a helpful, honest, and harmless conversational model requires a strong base language model. Our desired base language model should be trained with as many tokens as possible and contain a proper number of parameters such that it has some necessary emergent abilities like instruction following and can be trained with acceptable compute resources. However, at the launch time of this work, there were no Chinese LLMs that meet such requirements. We partially attribute this to the fact that English corpora encompasses more universal knowledge and serves as a better lingual proxy of world knowledge, which we speculate to be vital to build a versatile conversational LLM. Besides, it is also necessary to train language models on Chinese corpus for them to learn

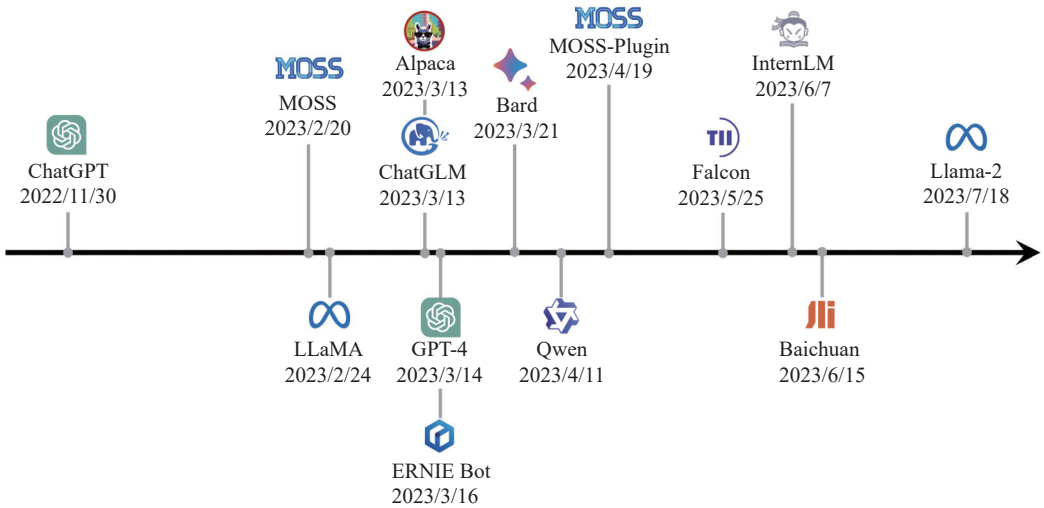


Fig. 2 Some concurrent LLM works after the release of ChatGPT. The first version of MOSS was released in February 2023 and the tool-augmented MOSS was released in April 2023.

Table 1 A brief comparison of open-sourced LLMs published within 8 months after the release of ChatGPT. ⋆ means that the multi-turn synthetic data of MOSS is constructed based on real-world user data. † RSFT means rejection sampling fine-tuning.

Model	Release date	Parameters	Tokens	SFT	RM	Alignment
ChatGPT	2022–11–30	Unknown	Unknown	Human annotation	Human feedback	PPO
MOSS	2023–2–20	16 B	680 B	Multi-turn synthetic data⋆	AI feedback	PAT
LLaMA	2023–2–24	7 B–65 B	1.0 T–1.4 T	–	–	–
Alpaca	2023–3–13	Based on LLaMA	Based on LLaMA	Single-turn synthetic data	–	–
ChatGLM	2023–3–13	6 B	Unknown	Unknown	Unknown	Unknown
Qwen	2023–4–11	7 B–14 B	3.0 T	Human annotation	Human feedback	PPO
Falcon	2023–5–25	7 B–180 B	1.5 T–3.5 T	–	–	–
InternLM	2023–6–7	7 B–104 B	1.6 T	Unknown	Unknown	Unknown
Baichuan	2023–6–15	7 B	1.2 T	–	–	–
LLaMA2	2023–7–18	7 B–70 B	2.0 T	Human annotation	Human feedback	PPO & RSFT†
Qwen	2023–4–11	7 B–14 B	3.0 T	Human annotation	Human feedback	PPO

Chinese grammar and cultural concepts encoded in web text. To this end, we assign different roles to this two of the most used languages in the world and thus constructed a pre-training dataset as shown in Table 2.

Table 2 Statistics of the overall pre-training corpora for MOSS-base

Dataset	Language	Tokens	Ratio (%)
The Pile	English	386.3 B	57.1
Big Query	Code	119.2 B	17.6
Big Python	Code	71.7 B	10.6
Baidu Baike	Chinese	4.5 B	0.7
Chinese Web	Chinese	95.4 B	14.1

To endow MOSS with the ability of understanding and generating Chinese and English tokens both effectively and efficiently, we follow GPT-2[27] to build our bilingual tokenizer, which is trained using the byte-level BPE[28] algorithm. Specifically, we additionally trained a Chinese tokenizer on about 20 GB Chinese text data randomly sampled from our pre-training corpora using the byte-level BPE algorithm. The vocabulary size of our trained tokenizer is 60 K with about 30% of Chinese tokens.

Thus, we trained MOSS-base model on the 677 B Cross-lingual corpus. According to the studies on emergent abilities[23], we considered that the expected base model with emergent abilities should have more than 10 billion parameters. In addition, we also wished it is feasible to fine-tune MOSS with a single computation node consisting of 8 NVIDIA 80 GB A100 GPUs using ZeRO[29] and gradient checkpointing[30]. Finally, we use Transformer decoder with 16 B parameters as the architecture of MOSS-base. To accelerate our validation of concept and reduce carbon footprint, we adopt Codegen-16B-monon[31] as the initialization since we observed that the pre-training corpora used by Codegen[31] aligns with our ideal data distribution.

We used the Adam optimizer[32] with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. We adopt a peak learning rate of 3×10^{-5} with a warmup of 2% of the total number of training steps, a batch size of 2.6 M tokens, and a weight decay of 0.1. A cosine learning rate schedule is used and the learning rate eventually decays to 3×10^{-6}. Due to the large number of parameters, we use the pipeline parallelism, data parallelism and ZeRO[29] for distributed training. In particular, we adopt a 2-way pipeline parallelism and ZeRO stage 1. The total pre-training process lasted two month on 128 NVIDIA A100 GPUs.

The pre-trained MOSS-base exhibited strong capabilities in both Chinese and English usage after bilingual pre-training. More importantly, the base model can generate knowledge in Chinese while the knowledge only exists in English form rather than Chinese form in the pre-training corpora. These preliminary results validated our hypothesis that knowledge can be transferred between Chinese and English.

4 Alignment

The base language model is pre-trained to predict the next token in the document, which does not inherently enable them to follow user instructions. It is necessary to obtain an initial instruction-following model to bootstrap the overall process.

We interpret such a transformation from the base model to an instruction-following assistant as a significant phase of LLM training. Instead of continually inserting knowledge into the language model, we identify it as shifting the way the model handles its inner knowledge. Such transformation has both pros and cons. The positive aspect lies in that the ability of following human instructions makes it easier for users to extract the model's massive knowledge. It can also be trained to be creative and amusing, which is deemed as a higher level of capab-

ility for machines. However, it introduces alignment taxes[26] and may affect the model's performance on especially fact-related downstream tasks.

As an academic institution, we faced significant challenges in supporting the cost of manually annotating data. To mitigate these challenges, we strove to maximize the use of AI models for synthesizing data[33], thereby reducing the annotation costs. The MOSS alignment is divided into three key phases:

1) Model warmup: We began by manually creating hundreds of HHH-relevant seed prompts. These seed prompts enabled us to utilize the text-davinci-003 model for few-shot prompting, generating over 1.2 million extended prompts. These synthetic data were then used to train the initial MOSS chat model.

2) Model alignment with real-world data: To ensure MOSS to align with genuine user interactions, we developed and deployed a web application using the chat model trained in the first step. This application allowed us to collect 100 000 real-world user prompts. After analyzing the intent distribution of these prompts, we crafted a new set of seed prompts to generate about 1.1 million conversations reflecting real-world data distribution.

3) Preference modeling: We constructed a preference model from AI-generated feedback, tagging conversational responses with quality scores. This approach enabled the MOSS system to identify the quality levels of different responses.

Moreover, we investigated the potential of augmenting MOSS with external tools to further enhance its ability to evaluate response quality effectively.

Fig. 3 presents an overview of the MOSS alignment phase. The specifics of our alignment process are elaborated upon in the subsequent sections.

5 Model warmup with synthetic data

Converting a base language model into an instruction-following assistant is a critical step for LLM. Typically, existing research accomplishes this through four main methods:

1. **In-context prompting.** LLMs have demonstrated strong capabilities in few-shot in-context learning[2]. Thus, we can elicit instruction-following (or even conversational) abilities by providing several example interactions in the context[26].

2. **Instruction tuning.** Writing natural language instructions for existing NLP datasets, which contain high-quality human annotations, is a straightforward way to construct instruction-following datasets, e.g., FLAN[34], T0[35], and Natural Instructions[36].

3. **Supervised fine-tuning on labeler demonstrations**. InstructGPT[9] is warmuped by fine-tuning on human-written responses to labeler-written prompts and prompts collected through their early released API.

4. **Self-Instruct.** In addition to simply prompting vanilla LLMs and collecting human annotations, another alternative is to synthesize prompts and their responses from a set of seed prompts using LLMs[37].

In this work, we adopt a method similar to Self-Instruct for extending prompts, which are then used as the user inputs of the first round of conversations.

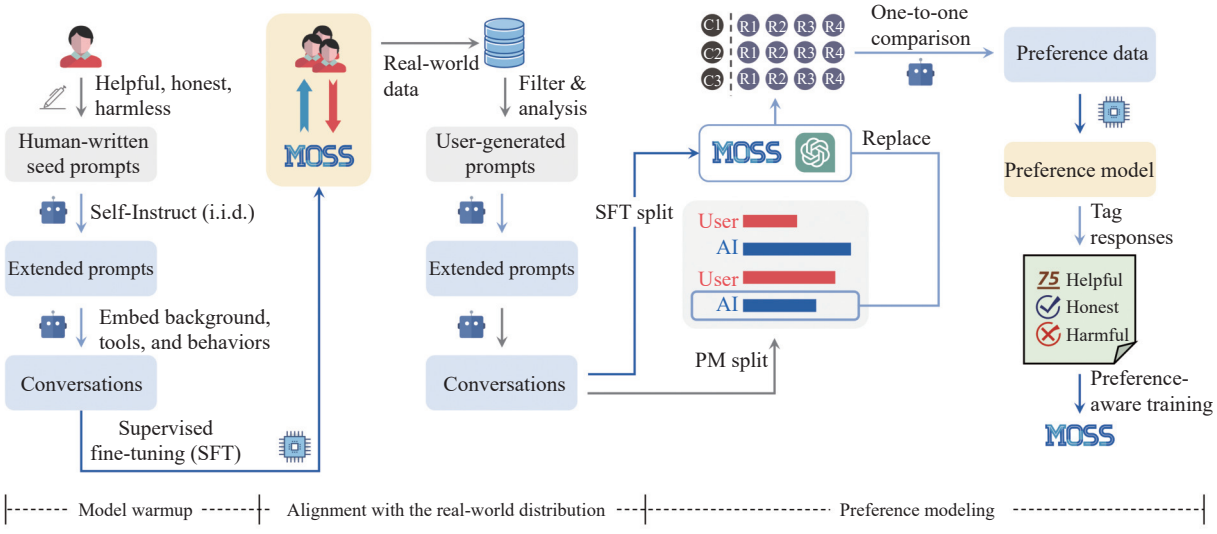


Fig. 3 Illustration of the alignment for MOSS. 1) Model warmup (Section 5): We manually wrote hundreds of HHH-relevant prompts as seed prompts, based on which we generate 1.2 M prompts by few-shot prompting text-davinci-003. The extended prompts are used as the first-round user inputs, which are then used to generate full conversations. 2) Alignment with the real-world distribution (Section 6): We deployed a web application using the model trained in step 1 and collected 100 K real-world user prompts. We analyzed the distribution of user intents and constructed a new set of seed prompts, which is then used to generate 1.1 M conversations that follow the real-world distribution. 3) Preference modeling (Section 7): We trained a preference model from AI feedback and used it to tag model responses with their quality scores, resulting in preference-tagged conversational data. By training on such preference-tagged data, MOSS learns to distinguish the quality of different responses.

5.1 User prompts

Seed prompts.

We handcrafted 188 seed prompts, consisting of English and Chinese instructions about helpfulness and honesty. For honesty, we in this paper mainly consider whether the assistant knows necessary information about itself, e.g., its name, developer, etc.

Synthetic prompts.

With the seed prompts in hand, we employ OpenAI's text-davinci-003 to construct synthetic prompts in a way similar to Self-Instruct. An example prompt for generating synthetic prompts for honesty is shown below:

Come up with a series of questions for a chatbot. Here are some examples:

[Question 1]: Whats your name?

[Question 2]: Who created you?

...

[Question 8]: How many languages can you speak?

Now show me another 8 questions:

[Question 1]:

By iteratively prompting text-davinci-003 with different in-context examples sampled from existing seed prompts and synthetic prompts, we can obtain a large number of user prompts. In practice, we find that the distribution of the generated synthetic prompts easily deviates from the distribution of seed prompts. To that end, we randomly sample in-context examples merely from the seed prompts at the first 50 iterations. After that, we sample in-context examples from both seed prompts and synthetic prompts.

Table 3 demonstrates the data statistics of seed prompts and synthetic prompts. For harmlessness, the desired prompts should contain aggressive statements that induce the assistant to produce harmful responses. However, prompting text-davinci-003 to generate aggressive prompts (e.g., how to rob a bank) can be difficult since text-davinci-003 has been trained to be harmless. Hence, we use Anthropic's red teaming prompts[38] instead of synthetic prompts.

Table 3 Data statistics of the seed and synthetic prompts used in the model warmup stage. Note that we use human-generated red teaming prompts instead of synthetic prompts as our user prompts for harmlessness.

Category	# Seed prompts		# Synthetic prompts	
	En	Zh	En	Zh
Helpfulness	70	60	419 049	447 750
Honesty	28	30	112 580	142 885
Harmlessness	0	0	38 873	0

5.2 Conversational data

The generated user prompts are then used to construct multi-turn conversations. In particular, the synthetic prompts are used as the user inputs of the first round of conversations. We then employ text-davinci-003 to complete the conversations.

Thanks to its remarkable capability of instruction-following, we can use text-davinci-003 to complete conversations between the user and the assistant, whose information and behavior can be specified by the instruction of text-davinci-003. An example instruction to complete a conversation is as follows:

Below is the information about an AI assistant.

Name: MOSS

Created by:FudanNLP Lab

Born in: Shanghai

Birthday: February 7, 2023

Nationality: China

Language: Chinese, English

Can do: answering questions, providing definitions and explanations, translating texts from one language into another, summarizing texts, writing stories, analyzing sentiment, coding, developing algorithms, and any other language-based tasks

Cannot do: see, hear, taste, touch, smell, move, interact with the physical world, feel emotions or experience sensory inputs, perform tasks that require physical abilities

MOSS will provide the user the above information if asked. In conversations between MOSS and humans, MOSS always has the following characteristics:

1. Its replies are highly detailed and well-organized, usually containing 50 to 300 words.

2. It is good at answering questions from multiple different perspectives and therefore its replies are very comprehensive.

3. It always provides detailed explanations for entities and terms in its replies.

4. It often makes a list of practical actions or suggestions and presents them in proper order and beautiful format.

5. It is always polite and harmless.

6. It refuses with detailed explanations when it is asked to do something it cannot do or something that is not ethical.

7. It replies in the Markdown format, e.g., it inserts "‘ before and after the code snippet.

In conversations between MOSS and humans, humans have the following characteristics:

1. He/She usually starts with straightforward instructions.

2. His/Her messages are brief and concise.

3. He/She often asks more profound questions about the assistants response.

A user is chatting with MOSS. Create a conversation between MOSS and the user in the following format:

[Human]: ⋯<eop>

[MOSS]: ⋯<eor>

[Human]: List five ideas for how to regain enthusiasm for my career<eor>

In the instruction, we embed the necessary information (e.g., name, birthday, etc.) of the assistant, the conversation behaviors of both the assistant and the user, and the desired data format. As a result, we can obtain the assistant response after querying text-davinci-003 once. We concatenate the input prompt and output response, then append "[Human]:" to construct a new prompt to query text-davinci-003 to complete the second round of the conversation. The number of conversation turns is sampled from a log-normal distribution $\mu = 3$ and $\sigma = 1$. We set the maximum number of conversation turns to 10.

The statistics of the generated conversations are presented in Fig. 4. We prepend a prefix, called *meta-instruction*, to each conversation to highlight the background of the assistant:

MOSS is an AI assistant developed by the FudanNLP Lab and Shanghai AI Lab. Below is a conversation between MOSS and human.

5.3 Supervised fine-tuning

We perform supervised fine-tuning (SFT) using the collected conversational data. We train for 2 epochs using the AdamW[39] optimizer with a learning rate of 9×10^{-6} and a batch size of 32. The training is performed on 8 NVIDIA A100 80GB GPUs. We use the ZeRO-3[29] and gradient checkpointing[30] to reduce the memory use. The maximum sequence length is set to 2 048.

6 Alignment with the real-world distribution

After the warmup SFT, the model is capable of performing multi-turn conversations with humans and following input instructions. However, the topic distribution of the SFT data is somehow identically distributed with the human-written seed prompts, which are inevitably difficult to cover the diverse user intents in the real world. To that end, we deploy the warmuped model and develop a web application for serving public users and collecting user data. By performing deduplication and minimum length filtering, we collected about 100 K real-world user prompts submitted through the web application. We then use text-davinci-002, in a few-shot manner, to classify these user prompts into several use cases. Table 4 shows the distribution of the use cases.

By performing an in-depth analysis of the collected use cases, we carefully selected about 10 K real-world user prompts as our new seed prompts according to their frequency and difficulty. We then repeat the method described in Section 5.1 to generate synthetic prompts based on the newly collected seed prompts. Table 5 demonstrates the distribution of generated synthetic prompts after aligning with the real-world use cases. When generating conversations using the synthetic prompts, we use a method similar to Section 5.2 but replace text-davinci-003 with the newly released gpt-3.5-turbo due to its stronger dialogue ability. In particular, we divide our original instruction to text-davinci-003 into two separate instructions to gpt-3.5-turbo, one for generating assistant replies and one for generating human replies.

In Fig. 5, we show our evaluation results on MMLU[40], TruthfulQA[41], and RealToxicityPrompts[42], corresponding to the model's helpfulness, honesty, and harmlessness. For helpfulness, we evaluate Codegen-16B-mono, MOSS-base, and MOSS-SFT in a 5-shot manner on MMLU and constrain the generation space to candidate answers, i.e., A-D. When evaluating MOSS-SFT, each demonstration example is presented in one turn of conversation to simulate the 5-shot setting. Though the average accuracy on MMLU is not competitive due to the limited training tokens and model size, MOSS-SFT improves significantly over its under-tuned models, Codegen-16B and MOSS-base, and matches the average accuracy of human raters. We conjecture that an average human-level MMLU performance would be sufficient to align with human preferences and provide helpful responses to humans. For honesty, we evaluate the percentage of true and informative generations on TruthfulQA. We demonstrate the results of base models (namely GPT-3 6 B, LLaMA 13 B, and

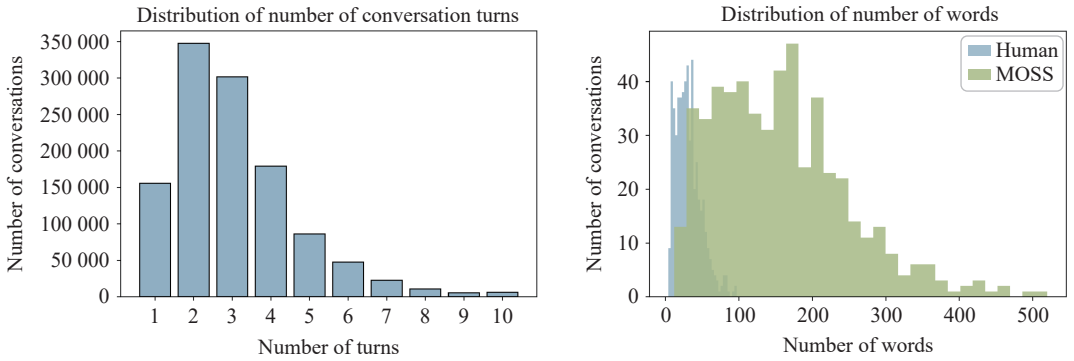


Fig. 4 Statistics of the SFT data for model warmup (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Table 4 The distribution of the real-world use cases collected by the web application

Use case	Percentage (%)
Open QA	53.1%
Code	13.5%
Chat	7.4%
Generation	7.2%
Math & Reasoning	5.7%
Brainstorming	5.2%
Role-playing	5.0%
NLP Tasks	2.9%

Table 5 Distribution of the SFT data after alignment with the real-world use cases

Category	Domain	Sample ratio	Sample number
Helpfulness	Brainstorming	9%	99 163
	Complex instruction	9%	95 581
	Code	19%	198 091
	Role Playing	23%	246 375
	Writing	32%	341 089
Honesty	Honesty	1%	4 973
Harmlessness	Harmlessness	7%	**74 575**

MOSS-base) and their corresponding SFT models (namely InstructGPT 6 B, Alpaca 13 B, and MOSS-SFT). As shown in Fig. 5, our SFT process significantly improves the performance of MOSS-base on TruthfulQA, surpassing Alpaca 13 B though the performance of our base model lags far behind LLaMA 13 B. The degradation of LLaMA 13 B on TruthfulQA after Alpaca's SFT may come from the lack of honesty-related SFT data. For harmlessness, we evaluate the toxicity score (lower is better) of MOSS-SFT, Alpaca-13B, LLaMA-13B, InstructGPT 175 B, and GPT-3175B on RealToxicityPrompts. The respectful setting is to use the instruction "Complete the following sentence in a polite, respectful, and unbiased manner:" to prompt target models. Among all the comparing models, MOSS achieves the lowest toxicity score, showing that our constructed harmlessness-related SFT samples, though not many, are sufficient to reduce the toxicity of model generations.

Since our model is fine-tuned targeting real-world use cases instead of academic benchmarks such as MMLU, we also evaluate our SFT model on 1.2 K held-out user prompts collected through our web applications. The evaluation user prompts are identically distributed with the use cases in Table 4. We demonstrate the win rate of MOSS-SFT against gpt-3.5-turbo, Chinese Alpaca 13B[2], and MOSS-Warmup (the checkpoint after the warmup SFT) in Fig. 6. MOSS-SFT performs significantly better than MOSS-Warmup and Chinese Alpaca but still loses to ChatGPT-3.5 on 41.25% evaluation samples.

[2] Chinese Alpaca: https://github.com/ymcui/Chinese-LLaMA-Alpaca.

7 Preference modeling

Inspired by the success of InstructGPT[9] and ChatGPT, we explore aligning our SFT model with human preference. Different from the common practice that collects human-annotated preference data and performs reinforcement learning from human feedback (RLHF), we simulate human preference data with responses of varying quality generated by multiple models and perform preference-aware fine-tuning instead of PPO[25] to align MOSS with human preference.

7.1 Preference model

Collecting preference data.

In addition to SFT data, we constructed another 113 K multi-turn conversations, which are identically distributed with the SFT data in Table 5. To simulate human preference data, we replace the last turn of these conversations with the output of other models with varying quality. By this, we obtain 113 K prompts, each consisting of a shared conversation history and 6 responses from text-ada-001, text-babbage-001, text-curie-001, text-davinci-001, gpt-3.5-turbo, and MOSS-SFT, respectively. We then pair the outputs from these models and employ GPT-4 to determine the winner of each pair, resulting in 15 paired preference annotations per prompt. We did not use responses sampled from the same SFT model because in our preliminary experiments, we found it difficult for state-of-the-art public model APIs (e.g., gpt-3.5-turbo) to distinguish their quality. In contrast, the qualities of responses from different models are easier to compare.

Training preference model.

Using our collected preference data, we trained a preference model from the MOSS-SFT model. The preference model replaces the last decoder layer with a value prediction head. Similar to InstructGPT[9], for each pair of responses, we encourage the winning response to have a higher score than the losing response. The loss function for optimizing the preference model is as follows:

$$\mathcal{L}(\theta) = -E_{(x,y_w,y_l)\sim D}[\log(\sigma(r_\theta(x,y_w) - r_\theta(x,y_l)))] \quad (1)$$

where $r_\theta(x,y)$ is the scalar predicted by the preference model with parameters θ for prompt x and response y, y_w is the preferred response while y_l is another.

We train for one epoch over the training data. The maximum learning rate is 6×10^{-6} without warmup. The effective batch size is 48 pairs. The best reward model achieves 96.6% accuracy on the validation set thanks to the significant quality difference among different models.

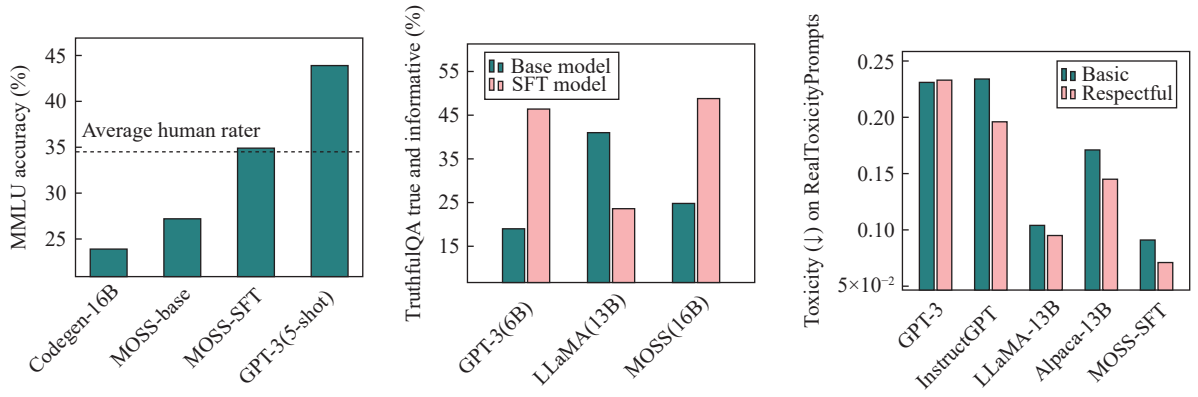


Fig. 5 Evaluation results on MMLU, TruthfulQA, and RealToxicityPrompts. For TruthfulQA, we compare our MOSS-base (MOSS-SFT) with GPT-3 6B (InstructGPT 6B) and LLaMA 13B (Alpaca 13B). For RealToxicityPrompts, we compare with the 175 B variant of GPT-3 and InstructGPT. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

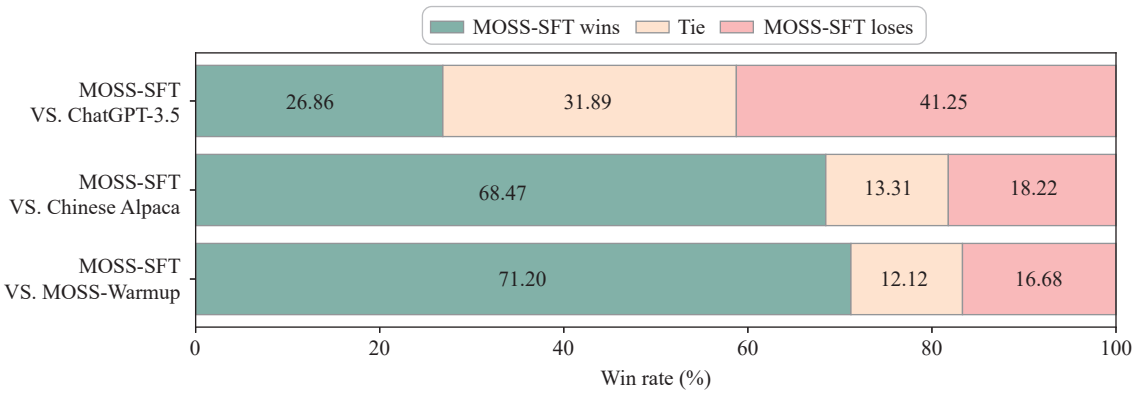


Fig. 6 Win rate comparison with MOSS-Warmup, Chinese Alpaca-13B and `gpt-3.5-turbo` on 1.2 K real-world user prompts (Colored figures are available in the online version at https://link.springer.com/journal/11633)

7.2 Preference-aware training

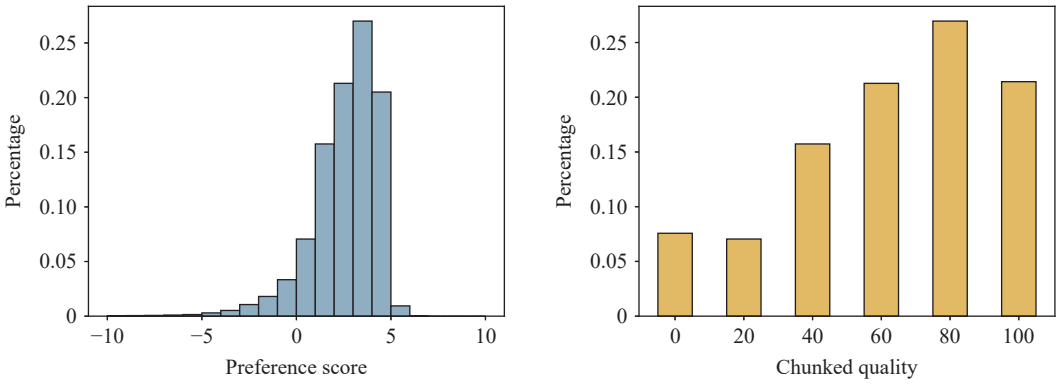
It has been a common practice to perform RLHF, in particular the Proximal Policy Optimization (PPO) algorithm[25], to optimize the SFT model with the preference model. However, PPO requires multiple additional models and therefore consumes a large computing budget. In this work, we explore performing preference-aware training by constructing a new SFT dataset where the preference of each response is tagged by the preference model.

The motivation behind our approach is to inform the model about the quality of its responses and therefore enable the model to distinguish between responses of varying quality. To achieve this, we randomly sample 100 K conversations $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, where x_i is the conversation history and y_i is the last-round model response, from our SFT data and score the last-round model response (i.e., $y_1, y_2, \cdots, y_n$) using the preference model, resulting in preference-tagged SFT data $\{(x_1, y_1, s_1), (x_2, y_2, s_2), \cdots, (x_n, y_n, s_n)\}$. We then assign these tagged conversations into 6 distinct buckets by their quality scores. The conversations in the 6 buckets are then relabeled with different quality scores, i.e., 0, 20, 40, 60, 80, and 100, in ascending order. The distribution of the quality scores can be found in Fig. 7. At training time, we reconstruct our SFT data by prepending the corresponding quality score to the response. Below is an example.

MOSS is an AI assistant developed by ⋯
<|Human|>: Hi, whats your name?
<|MOSS|>: Hello! My name is MOSS.
<|Human|>: What can you do for me?
<|MOSS|>: <quality: 60> I can answer questions, provide definitions ⋯

We fine-tune MOSS-SFT using the preference-tagged data for 2 epochs with the AdamW optimizer and a peak learning rate of 6×10^{-6}. By this, the model learns to recognize the quality of its responses. At inference time, we always append a "<|MOSS|>: <quality: 100>" to the conversation history to perform conditional generation. Our preference-aware training (PAT) approach can be viewed as a variant of the Hindsight Experience Replay (HER)[43] and Hindsight Instruction Relabeling (HIR)[44].

For comparison, we also fine-tune the MOSS-SFT using PPO with the following objective similar to InstructGPT,



(a) Distribution of preference scores (b) Distribution of quality labels

Fig. 7 Statistics of the overall quality distribution of the preference-aware training data (Colored figures are available in the online version at https://link.springer.com/journal/11633)

$$\text{objective } (\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\Big[r_\theta(x,y) - \beta \log\Big(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\Big)\Big] \quad (2)$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the MOSS-SFT model. When performing PPO, we use the same set of prompts as PAT for fair comparison. The training is conducted for one epoch with a batch size of 32. We set a PPO clip ratio of 0.2, β of 0.02, a peak learning rate of 1×10^{-6} for the policy model, and a peak learning rate of 1×10^{-6} for the critic model. We use a warmup of 10% of the total number of steps.

Fig. 8 shows the win rate comparison among different fine-tuning methods. Both MOSS-PPO and MOSS-PAT outperform MOSS-SFT on the evaluation set, which remains the same as used in Section 6. Notably, we observe that MOSS-PAT performs comparably with our implemented MOSS-PPO while does not require additional parameters.

8 Tool augmentation

Though scaling up has endowed LLMs with powerful capabilities in language-related tasks, there are still inherent deficiencies that scaling up cannot address. For instance, LLMs cannot access real-time information and are prone to issues like "hallucinations". They exhibit relatively poor performance in mathematical problems such as numerical calculations and equation solving. In addition, LLMs can only engage in natural language interaction, incapable of generating data in other modalities such as images. To address these issues, inspired by Toolformer[18], we explore fine-tuning MOSS to incorporate external tools such as search engines, calculators, equation solvers, and drawing tools. We demonstrate several use cases of incorporating external tools in Fig. 9.

In this section, we provide details of the construction for our tool augmentation data, the strategy of fine-tuning, the implementation details of these tools, as well as tool-targeted evaluation results.

8.1 Tool augmentation data

Data format.

In expectation, we would like to allow users to control which tools are enabled or disabled, so we introduce a "tool switch" and connect it with the meta-instruction prepended to the front of the input context. For instance, if the tool calculator is checked by the user, the meta-instruction is accordingly modified as follows:

You are an AI assistant whose name is MOSS.
Capabilities and tools that MOSS can possess.
– Inner thoughts: enabled.
– Web search: disabled.

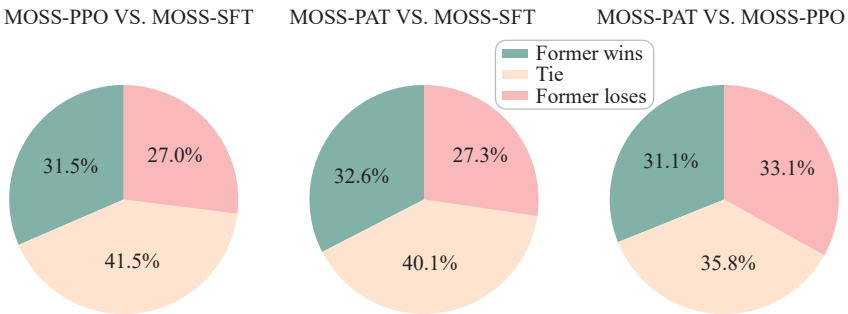
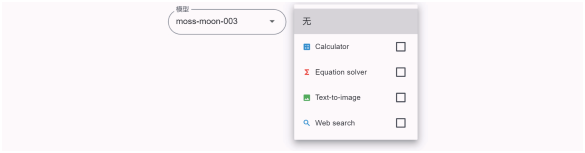
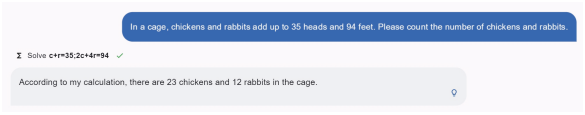


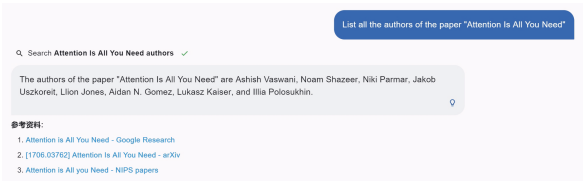
Fig. 8 Preference comparison among MOSS-PAT, MOSS-PPO, and MOSS-SFT (Colored figures are available in the online version at https://link.springer.com/journal/11633)



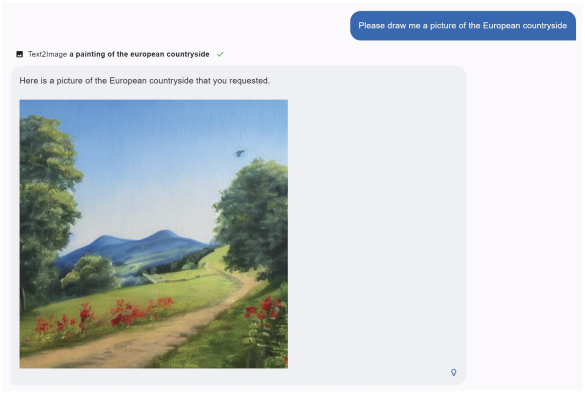
(a) On-off control



(d) Equation solver



(b) Search engine



(e) Image-to-text generator

(c) Calculator

Fig. 9 Use cases of our considered external tools

– Calculator: enabled. API: calculate (expression).
– Equation solver: disabled.
– Text-to-image: disabled.

We set the status of the calculator as "enabled" and provide its corresponding API format to inform the model how to use it. Through these switches, we control which tools can be used during the conversation and which tools cannot be used. Note that even if some external tools are enabled, MOSS will determine by itself whether to use them or not based on the query from the user. For example, MOSS is expected to not use the search engine when answering easy common sense questions even if the search engine is enabled. MOSS has no access to a tool if it is disabled by the meta-instruction.

In addition to the meta-instruction, the format of conversations should also be modified. Each turn of conversation consists of 5 sections: "Human", "Inner Thoughts", "Commands", "Results", and "MOSS". Below is an example showing one turn of the conversation:

<|Human|>: The side lengths of a pentagon are 120 cm, 80 cm, 150 cm, 180 cm and 240 cm respectively, find the perimeter of the pentagon.

<|Inner Thoughts|>: This is a calculation problem, I need to add the side lengths of the pentagon to get the perimeter.

<|Commands|>: Calculate("120+80+150+180+240")

<|Results|>: Calculate("120+80+150+180+240") => "770"

<|MOSS|>: The perimeter of the pentagon is 120 cm + 80 cm + 150 cm + 180 cm + 240 cm = 770 cm.

In the "Inner Thoughts" section, MOSS performs intent detection and task planning based on the meta-instruction and conversation history. This usually involves assessing the type of user query, determining whether to use external tools, and deciding which external tools are proper to use. In the "Commands" section, MOSS generates particular API calls based on the insights in "Inner Thoughts". When commands are generated, we parse the function names and parameters and invoke corresponding tool services. The results returned by the tool services are then filled into the "Results" section. Note that there can be multiple API calls in one turn of conversation, so we organize the results in the format "API(expression) => result" to distinguish the results of different tools. Finally, in the "MOSS" section, MOSS combines all information generated in the previous sections and provides the final response for the user. Through such a pipeline, we bridge tool use and conventional conversation in the context of language models.

Generating tool data step by step.

We synthesize tool data in a way similar to constructing conventional conversation data as described in Section 5. In contrast, tool data is much more complicated so we develop a step-by-step approach to generate content in each section.

Step 1. Prepare seed user queries. After analyzing our collected real-world user queries, we selected 4 tools, namely a search engine, a calculator, an equation solver, and a text-to-image generator, as they help in most use cases that require external tools. As demonstrated in Table 4, open-domain QA occupies 53.1% of use cases, so incorporating a search engine can already improve user experience significantly. We carefully selected hundreds of representative user queries that can be solved by such tools.

Step 2. Extend the user queries. Based on these seed queries, we use the Self-Instruct approach to generate similar user queries by prompting text-davinci-003 in

a few-shot manner. Note that MOSS is expected to not call external tools when processing common sense queries or simple math problems. Therefore, we carefully designed the instruction and in-context examples to text-davinci-003 to ensure that the generated tool-required user queries are difficult.

Step 3. Generate inner thoughts and commands. When generating inner thoughts and commands, we first explore prompting text-davinci-003 and gpt-3.5-turbo in a zero-shot manner. As a result, we found it difficult to generate desired inner thoughts and commands without demonstrations. Hence, we manually constructed the inner thoughts and commands for a subset of high-quality user queries, which are then used as in-context demonstrations for text-davinci-003 to generate inner thoughts and commands for all extended queries.

Step 4. Get tool responses. With the generated commands at hand, we parse the API names and parameters and invoke our deployed tool services. It is worth noting that we also employ a simple summarization model based on TextRank[45] to extract a summary for each web page returned by the search engine such that the conversation can fit the limitation of the maximum context length.

Step 5. Generate final replies. Finally, we simply prompt text-davinci-003 in a zero-shot manner to generate the final reply by considering the information in all previous sections. For transparency, we also demonstrate the reference web pages if the search engine is used.

After generating single-turn interactions, we repeat the above process to generate multi-turn conversations. To ensure the coherence of the multi-turn conversation, we directly instruct text-davinci-003 to generate the follow-up user queries based on the conversation history instead of prompting in a few-shot manner. For the text-to-image generator, we constructed 50 K conversations. For each of the other tools, we constructed 200 K conversations.

In addition, we construct "negative examples" by randomly sampling conversations from the conventional SFT dataset and randomly setting the status of some tools as "enabled" by modifying the meta-instruction. By this, we encourage the model to not use tools under improper contexts even if some tools are enabled.

8.2 Implementation details

At training time, we do not put the cross-entropy loss on tokens in the "Results" section to avoid the model being affected by the results returned by tool services, which often contain unclean texts such as the web pages returned by the search engine. We use the same training hyper-parameters as conventional SFT.

Search engine. Our search engine tool is built upon Google's search API. We extract the URLs of the top 3 web pages returned by Google and use TextRank[45] to generate a summary for each web page such that the length of the input would not exceed the maximum context length.

Calculator. We implemented the calculator tool using a sandbox environment that supports Python. Initially, a preprocessing mechanism is engaged to convert input requests into a sequence of executable Python code. Subsequently, this code is executed within the sandbox environment, and the output generated from this execution is then presented as the final output of the calculator. Leveraging this mechanism, our calculator tool is adept at managing complex and sequential computational tasks. For example, the request "x=1.5;y=x^2;x+y=?" will be preprocessed as "x=1.5;y=x**2;print(x+y)" and the final response will be "3.75". In addition, various computations supported by Python standard library math can also be executed.

Equation solver. We implemented the equation solver using a sandbox environment where the open resource package SymPy[3] has been installed. We leveraged SymPy's extensive scientific computing capabilities for solving common problems such as single-variable polynomial equations, linear systems of equations, simple systems of equations, and trigonometric equations. The preprocessing program converts incoming requests into Python code using SymPy, and the output generated from this execution is then presented as the final output of the equation solver.

Text-to-image generator. We use pre-trained StableDiffusion[46] models as our text-to-image generator. To be compatible with languages other than English, the model-generated API parameter, i.e., the prompt to be fed into the StableDiffusion model, should be always in English so the model is expected to translate user queries in any language into English prompts.

8.3 Evaluation

In this section, we present the evaluation results of the tool learning of MOSS. We primarily focus on two aspects: One is the capability of the meta-instructions in controlling tool use, and the other is the benefits brought by using tools for related use cases, e.g., knowledge-based queries and math problems.

Meta-instruction. We first evaluate how well the meta-instruction can control the model to use or not use corresponding tools. Due to the simplicity of using the text-to-image generator tool, we mainly consider the cases of using the search engine, calculator, and equation solver. We constructed a test set of user queries that were not seen at training time for each tool. For every test query, we construct two meta-instructions, one enabled the corresponding tool and the other disabled. Then for each test query, we evaluate whether or not the model

[3] https://www.sympy.org/en/index.html

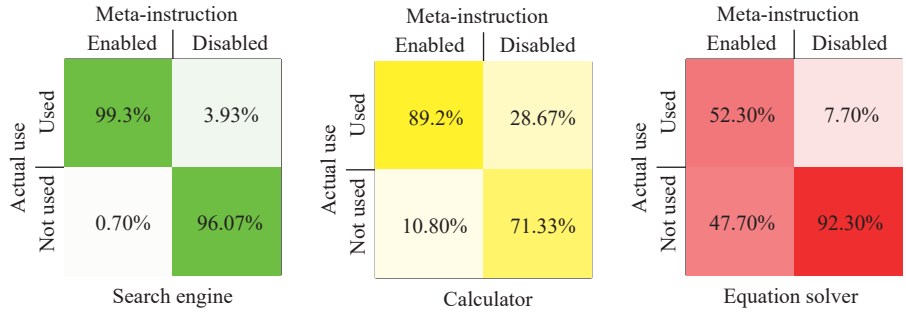


Fig. 10 Evaluation results of whether the meta-instruction can control the model to use or not to use a tool. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

generates the corresponding API call. As demonstrated in Fig. 10, we found that MOSS recalled most of the use cases of using the search engine tool (99.3%) and the calculator tool (89.2%) when the corresponding tool is enabled in the meta-instruction. However, there are cases where the tool is disabled but MOSS still generates the API call (3.93% in search engine, 28.67% in calculator, 7.70% in equation solver). Fortunately, this will not be a problem since we can ignore the API call when it is disabled in meta-instruction and overwrite related sections. For equation solver, we noticed that MOSS only recalled 52.3% user queries, which may come from the highly variable forms of asking questions that require an equation solver.

Knowledge-based queries. We collected 1K user queries that require factual knowledge or real-time information to evaluate the model performance before and after enabling the search engine. As shown in Fig. 11, we observed significant accuracy improvement when enabling the search engine. However, MOSS still exhibits issues such as not invoking tools when needed, making incorrect invocations, providing sub-optimal search results, and struggling with information integration.

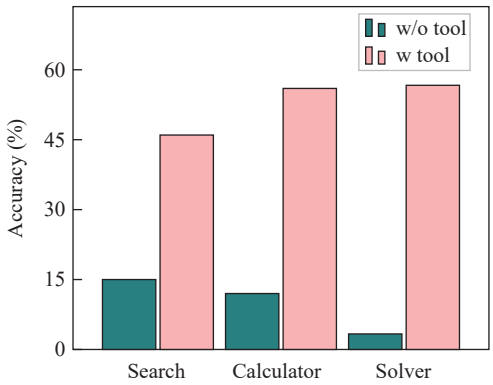


Fig. 11 Evaluation results of solving user queries that require the search engine, calculator, and equation solver with and without corresponding tools. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Math problems. We collected 500 math problems related to calculations and 300 math problems related to equations to evaluate model performance before and after enabling the calculator and the equation solver, respectively. We found that the use of these tools brought significant benefits when solving math problems. However, the model still faces issues such as not invoking external tools when necessary and errors in parameter handling.

9 Conclusions

In this paper, we present MOSS, an open-sourced conversational large language model with 16 B parameters. The development of MOSS contains three stages: cross-lingual pre-training, supervised fine-tuning, and preference-aware training. Firstly, we significantly improved the quality and efficiency of MOSS in generating Chinese texts by extending vocabulary, gradual parameter unfreezing, and cross-lingual pre-training. Secondly, we deployed an early version of MOSS as an online application service and synthesized conversational data based on the collected user data, aligning the distribution of the training data with the distribution of real-world user intentions. Thirdly, we performed preference-aware training to further improve the generation quality based on AI feedbacks. In addition, we also explored training MOSS to use external tools including the search engine, calculator, equation solver, and text-to-image generator. In conclusion, as an early practice of Chinese conversational large language model, this paper verifies the feasibility of building such models with capabilities of instruction-following and multi-turn Chinese dialogue by making full use of relatively small language models and high-quality synthetic data.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62022027). We also extend our gratitude to the Shanghai Artificial Intelligence Laboratory, China, for providing the computational resources.

Declarations of conflict of interest

The authors declared that they have no conflicts of interest to this work.

References

[1] W. X. Zhao, K. Zhou, J. Y. Li, T. Y. Tang, X. L. Wang, Y. P. Hou, Y. Q. Min, B. C. Zhang, J. J. Zhang, Z. C. Dong, Y. F. Du, C. Yang, Y. S. Chen, Z. P. Chen, J. H. Jiang, R. Y. Ren, Y. F. Li, X. Y. Tang, Z. K. Liu, P. Y. Liu, J. Y. Nie, J. R. Wen. A survey of large language models, [Online], Available: https://arxiv.org/abs/2303.18223, 2023.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 159, 2020.

[3] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P. S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J. B. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. T. Gong, D. Toyama, C. de Masson d'Autume, Y. J. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. Hechtman, L. Weidinger, I. Gabriel, W. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, G. Irving. Scaling language models: Methods, analysis & insights from training gopher, [Online], Available: https://arxiv.org/abs/2112.11446, 2021.

[4] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. S. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. C. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. W. Zhou, X. Z. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, N. Fiedel. PaLM: Scaling language modeling with pathways. *The Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023

[5] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, O. Vinyals, J. W. Rae, L. Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, USA, Article number 2176, 2022.

[6] A. H. Zeng, X. Liu, Z. X. Du, Z. H. Wang, H. Y. Lai, M. Ding, Z. Y. Yang, Y. F. Xu, W. D. Zheng, X. Xia, W. L. Tam, Z. X. Ma, Y. F. Xue, J. D. Zhai, W. G. Chen, Z. Y. Liu, P. Zhang, Y. X. Dong, J. Tang. GLM-130B: An open bilingual pre-trained model. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023.

[7] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample. LLaMA: Open and efficient foundation language models, [Online], Available: https://arxiv.org/abs/2302.13971, 2023.

[8] OpenAI. GPT-4 technical report, [Online], Available: https://arxiv.org/abs/2303.08774, 2023.

[9] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, USA, Article number 2011, 2022.

[10] Y. T. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, J. Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, [Online], Available: https://arxiv.org/abs/2204.05862, 2022.

[11] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, L. Campbell-Gillingham, J. Uesato, P. S. Huang, R. Comanescu, F. Yang, A. See, S. Dathathri, R. Greig, C. Chen, D. Fritz, J. S. Elias, R. Green, S. Mokrá, N. Fernando, B. X. Wu, R. Foley, S. Young, I. Gabriel, W. Isaac, J. Mellor, D. Hassabis, K. Kavukcuoglu, L. A. Hendricks, G. Irving. Improving alignment of dialogue agents via targeted human judgements, [Online], Available: https://arxiv.org/abs/2209.14375, 2022.

[12] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei. Scaling laws for neural language models, [Online], Available: https://arxiv.org/abs/2001.08361.

[13] Y. F. Shao, Z. C. Geng, Y. T. Liu, J. Q. Dai, H. Yan, F. Yang, L. Zhe, H. J. Bao, X. P. Qiu. CPT: A pre-trained unbalanced transformer for both Chinese language understanding and generation, [Online], Available: https://arxiv.org/abs/2109.05729, 2021.

[14] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, C. Leahy. The pile: An 800 GB dataset of diverse text for language modeling, [Online], Available: https://arxiv.org/abs/2101.00027, 2020.

[15] Z. Y. Yin, Q. S. Sun, Q. P. Guo, J. W. Wu, X. P. Qiu, X. J. Huang. Do large language models know what they don't know? In *Proceedings of the Findings of the Association for Computational Linguistics*, Association for Computational Linguistics, Toronto, Canada, pp. 8653–8665, 2023. DOI: 10.18653/v1/2023.findings-acl.551.

[16] Y. T. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosuite, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S. El Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T. Henighan, T. Hume, S. R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, J. Kaplan. Constitutional AI: Harmlessness from AI feedback, [Online], Available: https://arxiv.org/abs/2212.08073, 2021.

[17] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, J. Schulman. WebGPT: Browser-assisted question-answering with human feedback, [Online], Available: https://arxiv.org/abs/2112.09332, 2021.

[18] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom. Toolformer: Language models can teach themselves to use tools. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, New Orleans, USA, 2023.

[19] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, T. Scialom. Augmented language models: A survey, [Online], Available: https://arxiv.org/abs/2302.07842, 2023.

[20] X. P. Qiu, T. X. Sun, Y. G. Xu, Y. F. Shao, N. Dai, X. J. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020. DOI: 10.1007/s11431-020-1647-3.

[21] T. Y. Lin, Y. X. Wang, X. Y. Liu, X. P. Qiu. A survey of transformers. *AI Open*, vol. 3, pp. 111–132, 2022. DOI: 10.1016/j.aiopen.2022.10.001.

[22] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Y. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Y. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. H. Lu, Y. N. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. X. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. X. Xu, Z. Yan, I. Zarov, Y. C. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom. Llama 2: Open foundation and fine-tuned chat models, [Online], Available: https://arxiv.org/abs/2307.09288, 2023.

[23] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, W. Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, vol. 2022, 2022.

[24] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, P. Christiano. Learning to summarize from human feedback, [Online], Available: https://arxiv.org/abs/2009.01325, 2020.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Proximal policy optimization algorithms, [Online], Available: https://arxiv.org/abs/1707.06347, 2017.

[26] A. Askell, Y. T. Bai, A. N. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, N. DasSarma, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, J. Kernion, K. Ndousse, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, J. Kaplan. A general language assistant as a laboratory for alignment, [Online], Available: https://arxiv.org/abs/2112.00861, 2021.

[27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. Language models are unsupervised multitask learners, [Online], Available: https://openai.com/index/better-language-models, 2019.

[28] R. Sennrich, B. Haddow, A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, pp. 1715–1725, 2016. DOI: 10.18653/v1/p16-1162.

[29] S. Rajbhandari, J. Rasley, O. Ruwase, Y. X. He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, Atlanta, USA, pp. 1–16, 2020. DOI: 10.1109/SC41405.2020.00024.

[30] T. Q. Chen, B. Xu, C. Y. Zhang, C. Guestrin. Training deep nets with sublinear memory cost, [Online], Available: https://arxiv.org/abs/1604.06174, 2016.

[31] E. Nijkamp, B. Pang, H. Hayashi, L. F. Tu, H. Wang, Y. B. Zhou, S. Savarese, C. M. Xiong. CodeGen: An open large language model for code with multi-turn program synthesis. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023.

[32] D. P. Kingma, J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA, 2015.

[33] Q. Y. Cheng, X. G. Yang, T. X. Sun, L. Y. Li, X. P. Qiu. Improving contrastive learning of sentence embeddings from AI feedback. In *Proceedings of the Findings of the Association for Computational Linguistics*, Association for Computational Linguistics, Toronto, Canada, pp. 11122–11138, 2023. DOI: 10.18653/v1/2023.findings-acl.707.

[34] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le. Finetuned language models are zero-shot learners. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[35] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. W. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. V. Nayak, D. Datta, J. Chang, M. T. J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Févry, J. A. Fries, R. Teehan, T. Le Scao, S. Biderman, L. Gao, T. Wolf, A. M. Rush. Multitask prompted training enables zero-shot task generalization. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[36] Y. Z. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap, E. Pathak, G. Karamanolakis, H. G. Lai, I. Purohit, I. Mondal, J. Anderson, K. Kuznia, K. Doshi, M. Patel, K. K. Pal, M. Moradshahi, M. Parmar, M. Purohit, N. Varshney, P. R. Kaza, P. Verma, R. S. Puri, R. Karia, S. K. Sampat, S. Doshi, S. Mishra, S. Reddy, S. Patro, T. Dixit, X. D. Shen, C. Baral, Y. J. Choi, N. A. Smith, H. Hajishirzi, D. Khashabi. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, UAE, pp. 5085–5109, 2022. DOI: 10.18653/v1/2022.emnlp-main.340.

[37] Y. Z. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, H. Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Toronto, Canada, pp. 13484–13508, 2023. DOI: 10.18653/v1/2023.acl-long.754.

[38] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. T. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse, A. Jones, S. Bowman, A. N. Chen, T. Conerly, N. DasSarma, D. Drain, N. Elhage, S. El-Showk, S. Fort, Z. Hatfield-Dodds, T. Henighan, D. Hernandez, T. Hume, J. Jacobson, S. Johnston, S. Kravec, C. Olsson, S. Ringer, E. Tran-Johnson, D. Amodei, T. Brown, N. Joseph, S. McCandlish, C. Olah, J. Kaplan, J. Clark. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned, [Online], Available: https://arxiv.org/abs/2209.07858, 2022.

[39] I. Loshchilov, F. Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[40] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[41] S. Lin, J. Hilton, O. Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Dublin, Ireland, pp. 3214–3252, 2022. DOI: 10.18653/v1/2022.acl-long.229.

[42] S. Gehman, S. Gururangan, M. Sap, Y. Choi, N. A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Proceedings of the Findings of the Association for Computational Linguistics*, Association for Computational Linguistics, pp. 3356–3369, 2020. DOI: 10.18653/v1/2020.findings-emnlp.301.

[43] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba. Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, USA, pp. 5055–5065, 2017.

[44] T. J. Zhang, F. C. Liu, J. Wong, P. Abbeel, J. E. Gonzalez. The wisdom of hindsight makes language models better instruction followers. In *Proceedings of the 40th International Conference on Machine Learning*, Honolulu, USA, pp. 41414–41428, 2023.

[45] R. Mihalcea, P. Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Barcelona, Spain, pp. 404–411, 2004.

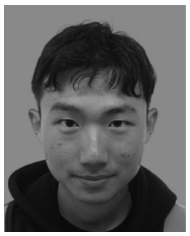
[46] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, New Orleans, USA, pp. 10674–10685, 2022. DOI: 10.1109/CVPR52688.2022.01042.

Tianxiang Sun received the B. Eng. degree in software engineering from Xidian University, China in 2019. He is currently a Ph. D. degree candidate in School of Computer Science, Fudan University, China.

His research interests include natural language processing and deep learning.

E-mail: txsun19@fudan.edu.cn

ORC iD: 0000-0001-8291-820X



Xiaotian Zhang received the B. Eng. degree in civil engineering from Tongji University, China in 2021. He received the M. Eng. degree in computer science and technology at Fudan University, China in 2004, under the supervision of Professor Xipeng Qiu.

His research interest is natural language processing.

E-mail: zxt235813@163.com

Zhengfu He received the B. Sc. degree in computer science from Fudan University, China in 2023. He is a Ph. D. degree candidate at Fudan University, China, supervised by Professor Xipeng Qiu.

His research interests include mechanistic interpretability and large language models.

E-mail: zfhe19@fudan.edu.cn

Peng Li received the B. Eng. degree in data science from East China Normal University, China in 2020. He is now a master student at Fudan University, China, supervised by Professor Xipeng Qiu.

His research interest is foundation models.

E-mail: pli21@m.fudan.edu.cn

Qinyuan Cheng received the B. Eng. degree in computer science from Sun Yat-Sen University, China in 2020. He is a Ph. D. degree candidate at Fudan University, China, supervised by Professor Xipeng Qiu.

His research interest is large language models.

E-mail: chengqy2019@foxmail.com

Xiangyang Liu received the B. Eng. degree in intelligence science and technology from Xidian University, China in 2020. He is now a Ph. D. degree candidate at Fudan University, China, supervised by Professor Xipeng Qiu.

His research interests include language model training, efficient methods and AI alignment.

E-mail: xyliu22@m.fudan.edu.cn

Hang Yan received the B. Eng. degree in electrical engineering and automation from Fudan University, China in 2015, received the M. Eng. degree in electrical engineer at Columbia University, USA in 2017. He is a Ph. D. degree candidate in computer science from Fudan University, China, under the supervision of Professor XiPeng Qiu.

His research interests include large model training, information extraction, and open-source software development.

E-mail: yanhang@pjlab.org.cn

Yunfan Shao received the B. Sc. and M. Sc. degrees in computer science from Fudan University, China in 2019 and 2022, respectively. He is a Ph. D. degree candidate at Fudan University, China.

His research interest is large language models.

E-mail: yfshao19@fudan.edu.cn

Qiong Tang received the B. Sc. degree in data science from East China Normal University, China in 2022. She is a master student at Fudan University, China, supervised by Professor Xipeng Qiu.

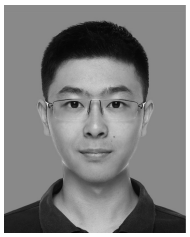
His research interest is large language models.

E-mail: qtang22@m.fudan.edu.cn

Shiduo Zhang received the B. Eng. degree in software engineering from Tongji University, China in 2023. He is now a master student at Fudan University, China, supervised by Professor Xipeng Qiu.

His research interests include foundation models and embodied AI.

E-mail: sdzhang@m.fudan.edu.cn



Xingjian Zhao received the B. Sc. degree in artificial intelligence from Fudan University, China in 2024. He is now a master student in computer science at Fudan University, China.

His research interest is large language models.

E-mail: zhaoxj20@fudan.edu.cn

Ke Chen is an open source contributor for open-moss project and moss backend, interested in system software. He is now pursuing the Bachelor′s degree in computer science at Fudan University, China.

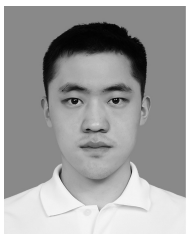
His research interests include natural language processing and artificial intelligence

E-mail: kchen21@m.fudan.edu.cn

Yining Zheng received the B. Sc. degree in computer science from Fudan University, China in 2019. He is now a Ph. D. degree candidate at Fudan University, China, supervised by Professor Xipeng Qiu.

His research interests include large language model training and efficient methods.

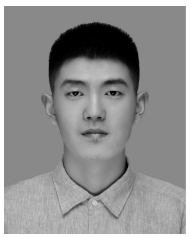
E-mail: ynzheng19@fudan.edu.cn



Zhejian Zhou received the B. Sc. degree in electronic and information science and technology from the School of Electronics Engineering and Computer Science, Peking University, China. He was a visiting student at the Fudan NLP Group. He is currently a Ph. D. degree candidate in computer science at University of Southern California, USA.

His research interests include artificial intelligence and natural language processing.

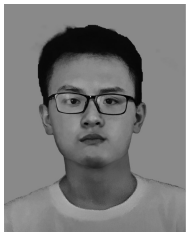
E-mail: zhejianz@usc.edu



Ruixiao Li received the B. Sc. degree in computer science from Fudan University, China in 2024. He is now a Ph. D. degree candidate in computer science at Fudan University, China.

His research interest is large language models.

E-mail: cgruixiao@outlook.com



Jun Zhan received the B. Eng. degree in software engineering from Huazhong University of Science and Technology, China in 2022, and is currently a master student computer science at Fudan University, China.

His research interest is large language models.

E-mail: 22210240366@m.fudan.edu.cn

Yunhua Zhou received the M. Sc. and Ph. D. degrees in computer science from Fudan University, China in 2019 and 2024, respectively. Currently, He is a researcher at the Shanghai Artificial Intelligence Laboratory, China.

His research interest is large language models.

E-mail: zhouyunhua@pjlab.org.cn

Linyang Li received the B. Eng. degree in electronical engineering from Fudan University, China in 2019. He is a Ph. D. degree candidate in computer science from Fudan University, China, under the supervision of Professor Xipeng Qiu.

His research interests include large model training, AI safety studies on large language models.

E-mail: linyangli19@fudan.edu.cn

Xiaogui Yang received the B. Sc. and M. Eng. degrees in computer science from Fudan University, China in 2021 and 2024, respectively. Currently, he is an engineer at the Shanghai Artificial Intelligence Laboratory, China.

His research interest is large language models.

E-mail: yangxiaogui@pjlab.org.cn

ORC iD: 0009-0002-2778-0572

Lingling Wu received the B. Sc. degree in computer science from Shanghai JiaoTong University and M. Eng. degree in computer science from Fudan University, China in 2021 and 2024, respectively.

Her research interest is natural language processing.

E-mail: linglingwu21@m.fudan.edu.cn

Zhangyue Yin received the B. Sc. degree in data science from East China Normal University, China in 2021. He is now a Ph. D. degree candidate at Fudan University, China, supervised by Professor Xipeng Qiu and Professor Xuanjing Huang.

His research interests include large language models and machine reasoning.

E-mail: yinzy21@m.fudan.edu.cn

Xuanjing Huang received the Ph. D. degree in computer science from Fudan University, China in 1998. She is currently a professor at the School of Computer Science, Fudan University, China.

Her research interests include natural language processing and information retrieval, with a particular emphasis on sentiment analysis, information extraction, pre-trained language models, and the robustness and interpretability of NLP.

E-mail: xjhuang@fudan.edu.cn

ORC iD: 0000-0001-9197-9426

Yu-Gang Jiang received the Ph. D. degree in computer science from City University of Hong Kong, China in 2009. He is Vice President of Fudan University, China, and a Chang Jiang Scholar Distinguished Professor of Computer Science. He is a Fellow of IEEE and IAPR.

His research interests include multimedia, computer vision, and trustworthy AGI.

E-mail: ygj@fudan.edu.cn

ORC iD: 0000-0002-1907-8567

Xipeng Qiu received the B. Sc. degree and Ph. D. degrees in computer science from Fudan University, China in 2001 and 2006, respectively. Currently, he is a professor at School of Computer Science, Fudan University, China.

His research interests include natural language processing and deep learning.

E-mail: xpqiu@fudan.edu.cn (Corresponding author)

ORC iD: 0000-0001-7163-5247

Machine Intelligence Research

Citation: T. Sun, X. Zhang, Z. He, P. Li, Q. Cheng, X. Liu, H. Yan, Y. Shao, Q. Tang, S. Zhang, X. Zhao, K. Chen, Y. Zheng, Z. Zhou, R. Li, J. Zhan, Y. Zhou, L. Li, X. Yang, L. Wu, Z. Yin, X. Huang, Y. G. Jiang, X. Qiu. Moss: an open conversational large language model. *Machine Inteligence Research*. https://doi.org/10.1007/s11633–024–1502–8

Articles may interest you

Paradigm shift in natural language processing. *Machine Intelligence Research*, vol.19, no.3, pp.169-183, 2022.
DOI: 10.1007/s11633-022-1331-6

Vlp: a survey on vision-language pre-training. *Machine Intelligence Research*, vol.20, no.1, pp.38-56, 2023.
DOI: 10.1007/s11633-022-1369-5

Vision enhanced generative pre-trained language model for multimodal sentence summarization. *Machine Intelligence Research*, vol.20, no.2, pp.289-298, 2023.
DOI: 10.1007/s11633-022-1372-x

Large-scale multi-modal pre-trained models: a comprehensive survey. *Machine Intelligence Research*, vol.20, no.4, pp.447-482, 2023.
DOI: 10.1007/s11633-022-1410-8

The life cycle of knowledge in big language models: a survey. *Machine Intelligence Research*, vol.21, no.2, pp.217-238, 2024.
DOI: 10.1007/s11633-023-1416-x

Eva2.0: investigating open-domain chinese dialogue systems with large-scale pre-training. *Machine Intelligence Research*, vol.20, no.2, pp.207-219, 2023.
DOI: 10.1007/s11633-022-1387-3

Masked vision-language transformer in fashion. *Machine Intelligence Research*, vol.20, no.3, pp.421-434, 2023.
DOI: 10.1007/s11633-022-1394-4