SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2025, Vol. 68, Iss. 9, 192103:1–192103:21
https://doi.org/10.1007/s11432-023-4128-3

OpenBA: an open-sourced 15B bilingual asymmetric Seq2Seq model pre-trained from scratch

Juntao LI, Zecheng TANG[†], Yuyang DING[†], Pinzheng WANG[†], Pei GUO, Wangjie YOU, Dan QIAO, Chenyu WANG, Wenliang CHEN, Guohong FU, Qiaoming ZHU, Guodong ZHOU[*] & Min ZHANG[*]

School of Computer Science and Technology, Soochow University, Suzhou 215006, China

Received 5 October 2023/Revised 29 Febrary 2024/Accepted 10 June 2024/Published online 19 August 2025

Abstract Large language models (LLMs) with billions of parameters have demonstrated outstanding performance on various natural language processing tasks. This report presents OpenBA, an open-sourced 15B bilingual asymmetric Seq2Seq model, to contribute an LLM variant to the Chinese-oriented open-source model community. We enhance OpenBA with effective and efficient techniques as well as adopt a three-stage training strategy to train the model from scratch. Our solution can also achieve very competitive performance with only 380B tokens, which is better than LLaMA-70B on the BELEBELE benchmark, BLOOM-176B on the MMLU benchmark, and GLM-130B on the C-Eval (hard) benchmark. This report provides the main details to pre-train an analogous model, including pre-training data processing, bilingual Flan data collection, the empirical observations that inspire our model architecture design, training objectives of different stages, and other enhancement techniques. Additionally, we also provide the fine-tuning details of OpenBA on five downstream tasks. We have refactored our code to follow the design principles of the Huggingface Transformers Library, making it more convenient for developers to use, and released checkpoints of different training stages at https://huggingface.co/openBA. More details of our project are available at https://github.com/OpenNLG/openBA.git.

Keywords open-source, Seq2Seq model, bilingual large language model, Flan data collection, large-scale training

Citation Li J T, Tang Z C, Ding Y Y, et al. OpenBA: an open-sourced 15B bilingual asymmetric Seq2Seq model pre-trained from scratch. Sci China Inf Sci, 2025, 68(9): 192103, https://doi.org/10.1007/s11432-023-4128-3

1 Introduction

The scaling law [1–4] of language models has brought unprecedented success. These large language models pre-trained on massive textual data demonstrate enormous superiority over previous paradigms for various fields and even have obtained newly emergent capabilities. Though very powerful and developing rapidly, these models at scale are still far from perfect or satisfactory for most of the real-world usages. To advance the development of LLMs, the open-source community has made great efforts to provide strong and publicly accessible LLMs, covering different data sources, architectures, language modeling objectives, training pipelines, model scales, and language of expertise, e.g., BLOOM [5], LLaMA [4,6], FlanT5 [7], AlexaTM [8].

As for Chinese, the open-source community has also released many large language models either by pre-training from scratch, e.g., GLM [9], Baichuan [10] or conducting further fine-tuning on existing open-sourced multilingual models, e.g., Huatuo [11], Luotuo [12], Phoenix [13], Chinese-LLaMA [14], MOSS [15]. These publicly available LLMs provide researchers and developers with strong general language models (i.e., the framework used by GLM [16]) and different decoder-only variants, but leaving the encoder-decoder framework (e.g., Flan-T5 [7]) under-explored, which has been proven universally effective for different prompt settings (zero-shot, few-shot, and chain-of-thought) [17] and various tasks (e.g., language understanding, commonsense reasoning, question answering, information retrieval, and multi-turn chit-chat conversation) [18,19].

* Corresponding author (email: gdzhou@suda.edu.cn, minzhang@suda.edu.cn)
† These authors contributed equally to this work.

To fill this blank, we contribute an open-sourced 15B bilingual asymmetric Seq2Seq model (OpenBA) pre-trained from scratch, providing not only the model checkpoints but also the data collection and processing details to construct pre-training data and bilingual Flan collection from openly accessible data resources (e.g., Common Crawl, the Pile corpus, C-Book), the motivations and empirical observations for the model architecture design, and the key details of other enhancement strategies. Concretely, our collected pre-training data consists of balanced English and Chinese tokens to make the Chinese language modeling benefit from high-quality English data. Since it is difficult to construct a Flan-like Chinese collection that covers extensive tasks and settings from open resources, we incorporate more English data sampled from the Flan collection in our bilingual-Flan corpus. Unlike the vanilla Flan-T5 [7] of a balanced encoder-decoder structure and the asymmetric deep-encoder shallow-decoder in AlexaTM [8], we utilize another asymmetric model structure, i.e., shallow-encoder deep-decoder to enhance the generation capability, which is motivated by our empirical study in Section 5.1. Our training process comprises three different stages, including the UL2 pre-training, length-adaptation, and Flan training. We also incorporate enhancement strategies in model architecture and training to improve model capability, training stability, and efficiency. Evaluations across different benchmarks (MMLU, CMMLU, C-Eval, SuperGLUE, BELEBELE, BBH) and tasks (e.g., understanding, reasoning, and generation) have demonstrated the effectiveness of our model in different settings (zero-shot, few-shot, held-in, and held-out). Though merely trained on 380B tokens, our model can surpass many representative models, e.g., LLaMA-70B on BELEBELE, BLOOM-176B on MMLU, ChatGLM-6B on CMMLU and C-Eval. Throughout the whole training process, OpenBA-15B produces around 6.5 tCO_{2eq} in total, which is much less than the LLaMA-7B model that costs 14 tCO_{2eq}.

Additionally, we further fine-tune the model on five downstream tasks, including bilingual multi-turn dialogue, code generation, instruction generation, tool retrieval, and named entity recognition, enabling OpenBA to become the expert model (OpenBA-X) for the downstream tasks. All the implementation details are open-accessible, not limited to data collection and processing, codes, model checkpoints, and evaluations.

2 Related work

2.1 Large language models

Benefiting from scaling law [1–3] and the growth of computational resources [20], the recent years have witnessed the remarkable evolution in the field of LLMs, which pushes the boundaries of various NLP tasks [4, 21–27]. The introduction of transformer model [28] is a notable turning point in the NLP field, as models based on such an architecture like GPT-2 [29] and T5 [30] have demonstrated outstanding performance across a wide range of NLP tasks by unifying the formulation of different tasks and scaling up the model size. This trend has continued with the advent of GPT-3 [22], which brings about groundbreaking advancements in scaling with its 175-billion-parameter model. Consequently, the research community has gradually recognized the benefits of LLMs, leading to a series of subsequent models following in rapid succession, such as Gopher [31], Megatron-Turing [32], Chinchilla [3], BLOOM [5], LLaMA [4, 6], ChatGPT [33, 34], Falcon [35], etc. These models have consistently advanced the frontiers in the NLP domain, promoting ongoing development and progress. However, in this process, two serious issues have gradually emerged. The first issue is the open-sourcing of LLMs. Due to concerns such as data sources and privacy [36], many LLMs are not available to the public or can only be accessed via limited or commercial APIs, e.g., ChatGPT [33] and PaLM [37], while the open-source alternatives have relative lower performance compared to their closed-source counterparts [38, 39]. Another issue is that, following the success of decoder-only models like GPT-3 and ChatGPT, the current focus of research mainly revolves around decoder-only architecture, while the investigation on large-scale encoder-decoder models, such as T5 [30] and AlexaTM [8], presents a relatively "under-explored area" in this field. Additionally, there is no clear consensus on whether encoder-decoder or decoder-only models hold an advantage over the others in terms of architectural superiority [18, 40]. In an effort to contribute to the open-source community and supplement the existing encoder-decoder models, we developed OpenBA, featuring an asymmetric encoder-decoder architecture. We collect and filter the pre-training data from open-accessible corpora. Additionally, we manually construct the Chinese Flan-like data from various publicly available annotated datasets and combine them with the English Flan corpus to obtain the bilingual Flan collection. We employ a stage-wise training strategy to optimize the model performance by utilizing these datasets. Our model achieves outstanding performance on multiple widely-used benchmarks, such as SuperGLUE [41] and C-Eval [42].

2.2 Instruction tuning

Instruction tuning, which involves the method of fine-tuning LLMs on an instruction dataset in a supervised manner, has played a crucial role in the significant advancements of LLMs in recent years [43]. Starting from the T5 model [30], which pioneers the concept of consolidating diverse NLP tasks as generative tasks. By employing task-specific prompts to guide the model, this method streamlines the process of applying LLMs to an extensive array of applications, laying the foundation for subsequent instruction tuning models such as FLAN [7, 44] and T0 [45], which further improve performance across diverse tasks by incorporating more task-specific instructions during the pre-training phase. An approach related to instruction tuning is chain-of-thought (CoT) prompting [46, 47], which enhances instructions with descriptions of intermediate reasoning steps, thereby boosting LLM performance [48–51]. At present, the open-source community offers a multitude of instruction datasets, such as Alpaca [52] and Dolly [53]. These instructions aim to enhance specific professional abilities of LLMs, such as code generation ability [54], or the general capabilities like commonsense reasoning skills [55]. However, the wide variety and inconsistent quality of these datasets pose challenges, with each dataset typically comprising a relatively small amount of data and focusing on a single language. In this work, we construct the BiFlan dataset, the first bilingual Flan dataset built upon the cleansed Flan data [17], containing various instruction types and tasks in English and Chinese language. Experimental results indicate that training on the BiFlan dataset can significantly enhance model performance on various strong benchmarks, such as MMLU [38] and CMMLU [39].

3 Methodology

This section presents the details of our OpenBA model, including our considerations and implementations in pre-training data collection and processing, bilingual Flan data construction, model architecture, training objectives and pipelines, as well as the model implementation and techniques.

3.1 Dataset collection

This part elaborates on the data collection and filtering process for each training stage: pre-training data for stages I and II (Section 3.1.1), and bilingual Flan (BiFlan) data for stage III (Section 3.1.2).

3.1.1 *Pre-training data collection and filtration*

Data sources. Considering that our primary target is to construct an open-source model, we collect our pre-training data from publicly accessible resources consisting of English, Chinese, and code data. Concretely, the English and code data are sampled from the Pile corpus [56][1)], which is a collection of 22 diverse high-quality subsets. The Chinese data is mainly collected from the Internet (i.e., a cleaned subset from Common Crawl[1)]), Chinese books (i.e., C-Book[1)]), News (i.e., news2016zh[1)]), Encyclopedias (i.e., baidu_baike[1)] and wiki2019zh_corpus[1)]), Comments (i.e., comments2019zh_corpus[1)]) and Laws (i.e., CAIL2018[1)]).

Filtration. Before mixing these different data components with a certain proportion, we filter them with both personal privacy protection and quality check strategies[2)], following [24]. Our designed filtration strategies are listed below.

- **Privacy filtering**. We removed all phone numbers, email addresses, and web links from the corpus to prevent potential privacy breaches.

1) Pile: https://pile.eleuther.ai/; Common Crawl: https://commoncrawl.org/; C-book: https://github.com/FudanNLPLAB/CBook-150K; news2016zh: https://github.com/CLUEbenchmark/CLUE; baidu_baike: https://github.com/BIT-ENGD/baidu_baike; wiki2019zh_corpus: https://github.com/CLUEbenchmark/CLUE; CAIL2018: https://github.com/thunlp/CAIL.

2) Since the Pile is a cleaned high-quality corpus, we directly sample English and code data from the Pile corpus without further filtration. Our filtration strategies are applied to the Chinese data.

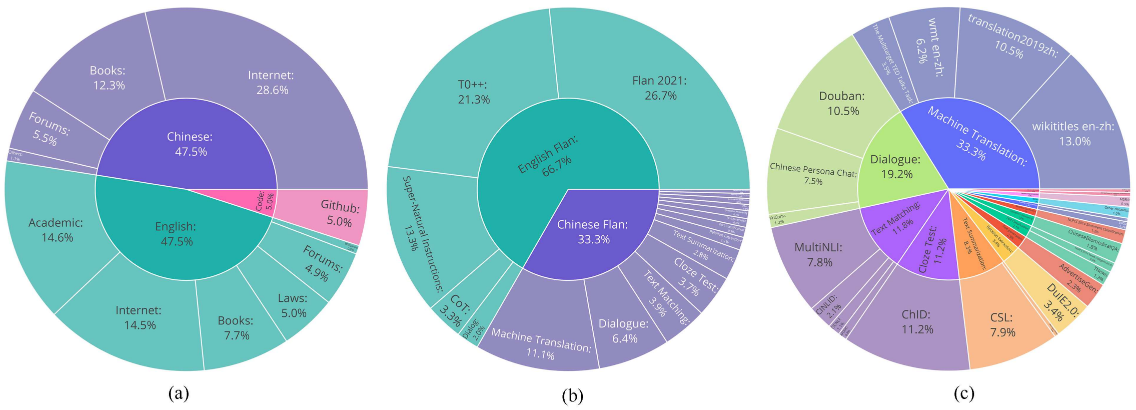


Figure 1 (Color online) The composition of data collection. (a) The composition ratio of the pre-training dataset; (b) the composition of the bilingual Flan dataset; (c) the finer-grained composition of the Chinese Flan dataset.

• **Deduplication**. Basically, we collect our data from different open-sourced datasets with disparate sources. Thus, we mainly conduct deduplication at document, character, and paragraph levels orderly. We treat each sample as a document and use a hash algorithm to remove redundant documents, i.e., retaining only unique documents. Similarly, we also leverage a hash algorithm with an extra sentence segmenter at the paragraph level to identify and remove repeated sentences or paragraphs (we treat consecutive 1–99 sentences as a paragraph). At the character level, we delete redundant characters and reduce the sequences of repeated characters to a single instance.

• **Language filtering**. We use polyglot[3] to detect the language of the text and only keep the texts with high confidence in either Chinese or English. We find it useful to filter out gibberish, especially for texts extracted from PDFs via OCR algorithms.

• **Internet data cleaning**. Data collected from the Internet often suffers from incompletions, unrecognizable characters, and web page tags. Thus, we filter out sentences with less than 10 words and remove unusual characters and HTML tags.

Mixing and statistics. Following Ref. [9], our pre-training data consists of the same proportion of Chinese and English tokens, in which we sample 190B English tokens[4] from the Pile corpus and 190B tokens from our filtrated Chinese data. We also sample 20B code tokens from the Pile corpus to make the overall percentages (5%) of code tokens resemble LLaMA [4] (4.5% code tokens). In total, our pre-training dataset is a mix of 190B English tokens, 190B Chinese tokens, and 20B code tokens. Following our strategies, one can construct a pre-training dataset with trillion tokens. Nevertheless, we only collect 400B tokens for pre-training due to our limited computation resources. Figure 1(a) shows the detailed composition of the pre-training dataset.

3.1.2 *Bilingual Flan data collection*

English Flan data collection. The Flan collection [7,17] serves as a foundational dataset for effective instruction tuning, encompassing more than 1800 tasks. We follow the official guidelines to collect and process the English Flan collection with two steps, i.e., downloading five sub-mixtures from the Flan collection and then combing them according to the specified mixture rates[5].

Chinese Flan data collection. Many open-source Chinese instruction datasets are derived from English translations or generated by ChatGPT using various prompts [33,34], which can lead to translation inaccuracies and incorrect responses. Thus, the quality and quantity of existing Chinese instruction corpora are often inadequate. To tackle these challenges, we build our own Chinese instruction corpus. More concretely, we collect 44 different Chinese tasks with a total of 50 million data entries, in which the data sources include various competitions, academic papers, and open-source projects. The distribution of the Chinese Flan dataset is shown in Figure 1(c), and we list all the data sources in the Supporting Information. For each task, we manually design the Chinese instructions.

3) https://github.com/aboSamoor/polyglot.

4) These English tokens exclude code data but inevitably contain a small percentage of non-language tokens (e.g., 1.24% of math data) since we randomly select samples based on the original proportion of the Pile corpus except for code data.

5) https://github.com/google-research/FLAN/tree/main/flan/v2.

Table 1 Model settings for OpenBA, where #Param. denotes the number of model parameters. We share the embedding weights between the encoder and decoder, which are not repeatedly counted.

Encoder	Decoder	Attn heads	d_{model}	d_{ff}	#Param. (B)
12	36	40	4096	16384	14.6

Table 2 Configurations for each training stage, where #Tokens represents the number of consumption tokens in each stage.

Stage	Strategy	Encoder context	Decoder context	Batch size	#Tokens (B)
I	UL2 pre-training	570	380	4096	300
II	Length-adaptation	1024	1024	1024	40
III	Bilingual Flan training	1024	256	1024	40

Bilingual data combination. Due to the smaller number of Chinese data compared to English data, we perform sampling within the English Flan datasets to ensure a balanced proportion between Chinese and English data. As illustrated in Figure 1(b), the bilingual Flan (BiFlan) dataset consists of 66.7% English Flan data and 33.3% Chinese Flan data. We also filter out samples whose lengths exceed the encoder's maximum length, ensuring the critical parts of instructions are not truncated.

3.2 Model architecture

Generally, the OpenBA model follows the standard encoder-decoder architecture like T5 [30]. However, it is worth noting that the encoder and decoder serve different roles, where the encoder provides strong comprehension capability, while the decoder offers generative ability [28], and existing works indicate that an encoder-decoder model with more encoder layers can achieve powerful performance [8]. To enhance generative capability and fill the gap of deeper decoder-based LLM, we also design another asymmetric structure, where the detailed model settings are given in Table 1.

Apart from leveraging the asymmetric shallow-encoder deep-decoder, we also incorporate the following improvement strategies into the model architecture.

• **Sandwich layer normalization**. To stabilize the training process, we adopt the sandwich layer normalization [57] by normalizing both the input of each transformer block and the output of each attention layer. We use the RMSNorm [58] for normalization.

• **Rotary embedding**. We apply the rotary embedding [59] that encodes the absolute position with a rotation matrix and meanwhile incorporates the explicit relative position dependency in self-attention instead of the relative positional embedding in T5.

• **SwiGLU activation function**. Inspired by LLaMA [4], we replace the original ReLU activation with the SwiGLU activation function [60], and set the dimension as $\frac{2}{3}4d$.

• **mT5 tokenizer**. For the bilingual setting, we use mT5 [61] tokenizer as it not only covers Chinese and English but also provides possibilities for further training and applications in other languages.

3.3 Training process and language modeling tasks

As shown in Figure 2, we adopt the stage-wise training strategy [62] that consists of UL2 [18] pre-training, length-adaptation, and Flan training stages [44], and set different context length and batch size for different stages (Table 2).

For all the stages, we adopt the span-denoising language modeling task as proposed in T5 [30] and BART [63]. More concretely, given a sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ containing n tokens, we corrupt it with mask sentinel $m_j = \{x_i\}_{i=p}^{k}$, where $p < k, 1 \leqslant p, k \leqslant n$. Then, the training objective is to recover the masked span, which can be formally written as:

$$\mathcal{L}(\boldsymbol{x}) = \log P(\boldsymbol{m} \mid \boldsymbol{x}_{\setminus \boldsymbol{m}}, \theta), \tag{1}$$

where $\boldsymbol{m} = \{m_j\}_{j=1}^{K}$ is the set of masked spans, $\boldsymbol{x}_{\setminus \boldsymbol{m}}$ denotes the corrupted sequence, and θ represents the model parameters. For OpenBA model, $\boldsymbol{x}_{\setminus \boldsymbol{m}}$ is fed to the encoder, which can retain a bidirectional receptive field, and $\boldsymbol{m}$ is predicted by the decoder. Next, we will introduce the aforementioned different stages more concretely.

Stage I: UL2 pre-training. Starting with the UL2 training strategy, we adopt a mixture of denoisers training strategy proposed by [18], exposing OpenBA to a diverse set of problems during the first pre-training stage. In this stage, the OpenBA model is fed with 300B tokens sampled from the pre-training corpus, and we employ three different training objectives which are listed below.

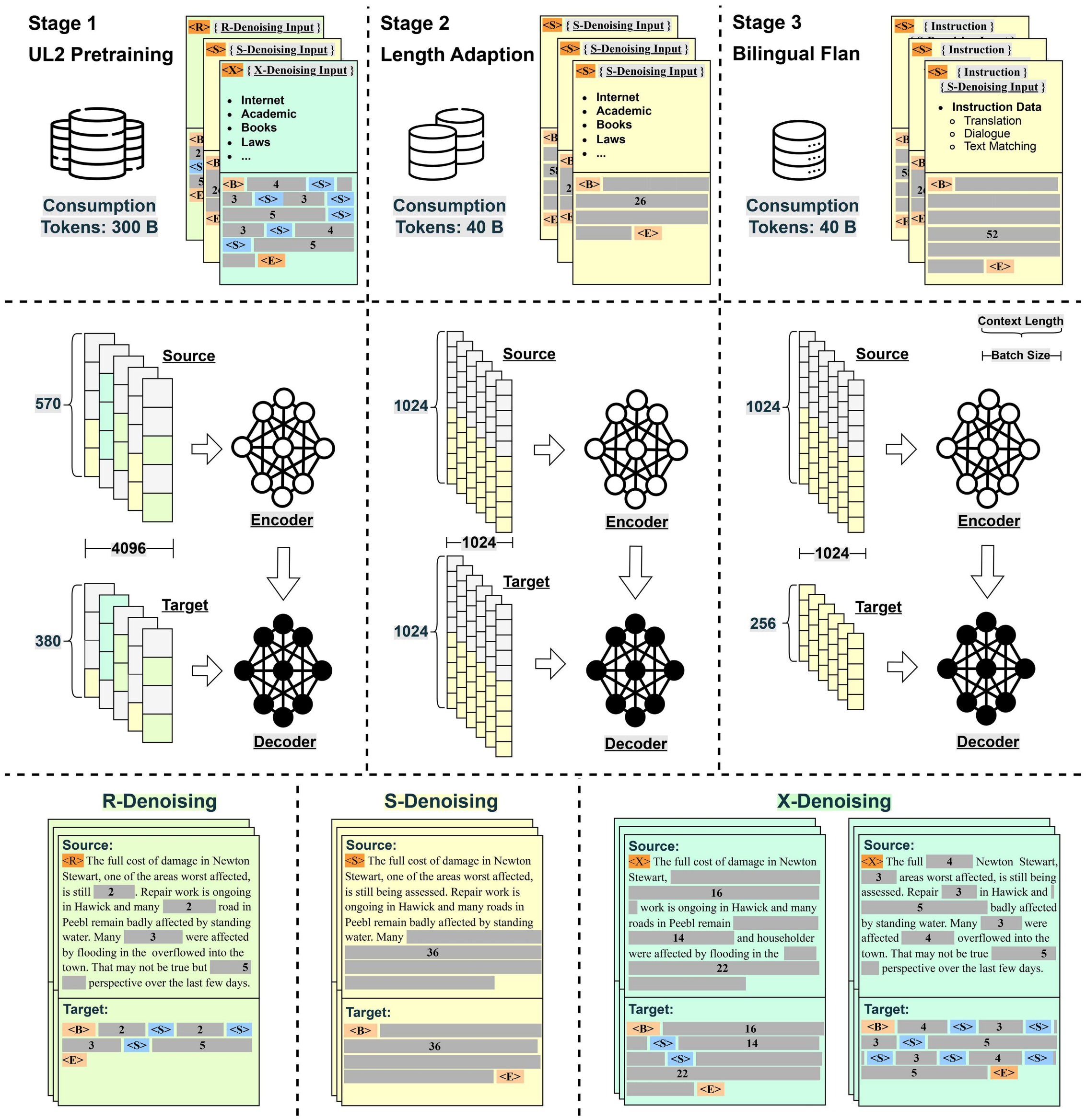

Stage 1
UL2 Pretraining
Consumption Tokens: 300 B
<R> { R-Denoising Input }
<S> { S-Denoising Input }
<X> { X-Denoising Input }
Internet
Academic
Books
Laws
...
Stage 2
Length Adaption
Consumption Tokens: 40 B
Stage 3
Bilingual Flan
Consumption Tokens: 40 B
{ Instruction }
Instruction Data
Translation
Dialogue
Text Matching
Source
Target
Encoder
Decoder
570
380
4096
1024
256
Context Length
Batch Size
R-Denoising
S-Denoising
X-Denoising
Source:
Target:
The full cost of damage in Newton Stewart, one of the areas worst affected, is still being assessed. Repair work is ongoing in Hawick and many roads in Peebl remain badly affected by standing water. Many were affected by flooding in the overflowed into the town. That may not be true but perspective over the last few days.

Figure 2 (Color online) Overview of the training process.

- **R-denoising**. Regular denoising is the standard span corruption that sets a range of 2 to 5 tokens as the masked span length and masks ratio about 15% of the input tokens. This denoising task is relatively simple since the span is short and efficient for the model to acquire knowledge embedded in the text.
- **S-denoising**. Sequence denoising aims to endow the model with generation capability, where the input text is split into two sub-sequences, and the model should predict the latter sequence conditioned on the first sequence. In the S-denoising setting, the model can acquire the generation ability.
- **X-denoising**. To bridge the gap between the R-denoising and S-denoising, X-denoising can be viewed as an extreme version of denoising, where approximately 50% of the input sequence is masked by increasing either the masked span length or the corruption rate. Such a denoising strategy simulates the situation where a model needs to generate long targets from a memory with relatively limited information.

We list the settings of these three denoising strategies in Table 3. It is worth noting that we conduct these denoising strategies from the instance level and prepend three special tokens before each corrupted sequence to prompt the current denoising task for OpenBA [18]. We uniformly sample a value based on μ as the masked span length for the R-denoising and X-denoising. For S-denoising, we limit each masked span to end at the end of the input text and allow only one masked span. Besides, we set encoder-decoder context length as 570/380 in this stage for sampling efficiency.

Table 3 Settings of three denoising strategies for the UL2 pre-training stage, where μ is the mean of the normal distribution, #Num represents the number of masked spans, and K is determined by the sequence length, span length, and corruption ratio.

Type	Span length (μ)	Corruption ratio (%)	#Num	Sentinel
R-denoising	{3, 8}	15.0	K	`<R>`
S-denoising	–	25.0	1	`<S>`
X-denoising	{3, 8, 64}/{64}	50.0/15.0	K	`<X>`

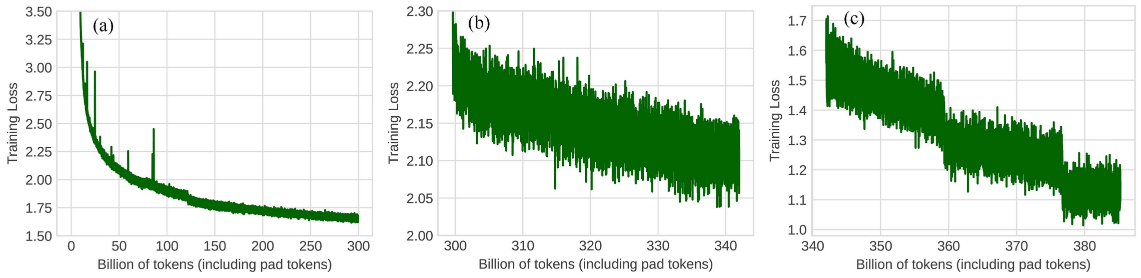


Figure 3 (Color online) Loss curves for each training stage. (a) Loss curve for the UL2 pretraining stage; (b) loss curve for the length adaptation stage; (c) loss curve for the bilingual flan training stage.

Stage II: length-adaptation. Considering the context length for the first pre-training stage is short, which may not support the long input and output formats of some tasks, such as in-context learning [64] and long text generation [65], we extend the encoder-decoder context length to 1024/1024 during the length-adaptation stage. During this stage, we utilize 40B tokens sampled from the pre-training corpus and ensure that there is no overlap between these data and the data from the previous stage. Additionally, we simply apply the S-denoising training objective and adjust the corruption ratio to 50%. We keep the special sentinel `<S>` before each corrupted text and decrease the batch size for training stability in this stage.

Stage III: bilingual Flan training. Inspired by the previous work [7], we apply Flan instruction training on the length-adapted OpenBA checkpoint. We still prepend the special token `<S>` before each text for the generation task and apply the constructed BiFlan dataset in this stage. In addition, we set the encoder-decoder sequence length as 1024/256 in this stage for sampling efficiency since we observe that most outputs of Flan datasets are short, i.e., less than 256 tokens.

3.4 Model implementation and techniques

We train OpenBA on a cluster with 4 nodes (8 $\times$ NVIDIA A100-SXM4-80GB GPUs), which are linked with the InfiniBand network [66] and interconnected through the NVLink system. The model has consumed nearly 400B bilingual tokens and achieved 1.2×10^{22} FLOPs (floating point of operations) in total. We implement our model based on the NVIDIA-Megatron framework[6)] and make several optimizations for training stabilization and inference efficiency. We plot the training loss for the aforementioned three stages in Figure 3, and list the techniques we have used below.

• **3D parallelism**. 3D parallelism [67] aims to scale and accelerate the training process of LLMs, which harnesses three core parallelism techniques, i.e., data parallelism, model parallelism (mp), and pipeline parallelism (pp). Considering the model size, the number of GPUs and the communication speed among GPUs, we settle on an optimal setting of *mp_size*=4 and *pp_size*=1, reaching 120 TFLOP/s per GPU.

• **Checkpoint activation**. Checkpoint activation is a technique designed to optimize memory usage during training. Instead of storing all intermediate layer activations, only certain ones are preserved. During back-propagation, the missing activations are recalculated, trading off additional computational efforts for memory savings. This strategy allows for the training of larger models even on GPUs with limited memory capacity. In fact, training a 15B model on 80GB GPUs becomes manageable in terms of memory. We specifically apply the checkpoint activation to the attention computation, which is relatively cost-effective to recompute. In practical deployment, we observe a significant improvement in GPU memory utilization, enhancing the overall system performance.

• **Distributed optimizer**. The distributed optimization approach offers an alternative for saving GPU memory, enabling the utilization of an increased batch size, albeit at the expense of communication

6) https://github.com/NVIDIA/Megatron-LM/.

Table 4 Overview of evaluation tasks/benchmarks and overlap between training set and testing set.

Task/Benchmark	Overlap between training set and testing set
Summarization, machine translation, text simplification paraphase, story generation, MMLU, CMMLU, BBH, C-Eval	\
SuperGLUE	English Flan training set (BoolQ, CB, RTE, ReCoRD, WSC, WiC, Copa, MultiRC)

burden among GPUs. By adopting the ZeRO method proposed by [68] and implementing the distributed optimization technique [67], we can increase the batch size, thereby enhancing the training speed.

• **Attention weights computation in FP32 precision**. During the softmax computation, particularly when handling large values, there exists a possibility of numerical overflow. Conducting this computation with FP32 precision mitigates this risk compared to using FP16 precision. The previous works [69] indicate that such an issue can easily take place when computing attention weights in FP16 precision. In the early training stage of OpenBA, we adopt half-precision calculation for all the model modules and often observe the phenomenon of loss collapsing. However, such an issue has been greatly alleviated when converting the attention weight calculations to full precision (FP32). Thus, we can empirically conclude that attention weight computation in FP32 precision can significantly enhance the stability of the training process.

• **Inference efficiency**. To accelerate the inference speed, we adopt the KV-cache technique and decrease the computation by pre-computing the rotary embeddings for all the positions.

4 Results

4.1 Evaluation settings

Task selection. Recent work has revealed that the risk brought by benchmark leakage can be reduced by diversified evaluation tasks across different types and complexities and data decontamination checking between training and testing data [70]. Following the empirical suggestion, we manually check whether there exist misused testing sets during the training phase and evaluate OpenBA from three different aspects: natural language understanding, natural language generation, and commonsense reasoning and different settings: Task held-in/held-out. Specifically, we evaluate the natural language understanding capability on the SuperGLUE [41] and BELEBELE [71] benchmark, natural language generation ability with five downstream tasks (summarization, machine translation, text simplification, paraphrase, and story generation), and commonsense reasoning ability on five benchmarks, including MMLU [38], CMMLU [39], BBH [72], and C-Eval [42]. Following the previous works [4, 22], we consider both the zero-shot and few-shot settings and strictly distinguish the dataset distribution of training and testing data. The illustration and the corresponding implementation of each setting are as follows.

• **Zero-shot**. We provide a textual description of the task for each testing sample, and the model will respond in an open-ended manner. Templates for all tasks are listed in the Supporting Information.

• **Few-shot**. We evaluate each example in the testing set by randomly selecting ℓ examples from the training set of each task as conditioning. In this paper, we set $\ell = 5$ as the default if not specified.

• **Task held-in/held-out**. We differentiate between the held-in and held-out settings based on whether the training data includes the task of the testing set. If the model has been trained on the training data corresponding to the testing task, it is viewed as held-in; otherwise, it is held-out [17]. Specifically, we carefully check the data overlap between training (i.e., bilingual Flan data) and testing sets[7)], and we list the overall dataset overlap in Table 4. Except for SuperGLUE, there is no overlap between the other datasets and the training data (held-out). For SuperGLUE, there exists some overlap between its subsets, e.g., BoolQ, and the English Flan training data.

We also apply the CoT technique for some tasks, and the corresponding templates are also shown in the Supporting Information. It is worth noting that we will specifically elaborate on the basic settings for each evaluation task and compare them to the models under the same settings. Additionally, we will evaluate the results using the officially recommended evaluation metrics and platforms whenever possible in all the experiments.

7) As fine-grained data contamination checking of pre-training data is computationally prohibitive, we manually check the overlap of data sources between Flan data and the correlated training data (if any) of each testing set.

Table 5 The number of parameters, consumed tokens, and training cost for the LLMs mentioned in the paper, where #Param. denotes the model parameters. We report the carbon emission according to the official statement, and calculate the carbon emission of OpenBA according to [73].

Model	#Param.	Tokens	GPU/TPU type	GPU hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT [27]	175B	180B	A100-80GB	809472	356 MWh	137
BLOOM [5]	176B	366B	A100-80GB	1082880	475 MWh	183
LLaMA [4]	7B	1.0T	A100-80GB	82432	36 MWh	14
LLaMA [4]	13B	1.0T	A100-80GB	135168	59 MWh	23
Baichuan [10]	7B	1.2T	A800	–	–	–
BatGPT [74]	15B	1.0T	–	–	–	–
MOSS [15]	16B	>700B	–	–	–	–
OpenBA	15B	380B	A100-80GB	38214	17 MWh	6.5

Model selection. Our selection criteria for comparative models are noteworthy. We focus on four criteria: firstly, we select the prevalent open-source models released around the same time as OpenBA, for instance, LLaMA; secondly, for Chinese tasks, we select the LLMs excelling in Chinese language like Baichuan; thirdly, we compare models with a similar number of parameters, such as BatGPT and MOSS; lastly, due to the relevant insufficient tokens in pre-training, we consider models with an equivalent computational cost, as represented by the total FLOPS of models like ChatGLM-6B.

4.2 Training cost analysis

All the models we compare are listed in Table 5, where we report their parameters, consumption tokens, training cost, and the corresponding carbon emissions, respectively. To calculate carbon emissions, we follow [73] and [4] by taking a PUE of 1.1 and a carbon intensity factor set at the national US average of 0.385 kg CO_{2eq} per kWh, and the formula is

$$tCO_{2eq} = MWh \times 0.385. \tag{2}$$

It is worth noting that, the training process of OpenBA is highly efficient and environmentally friendly. Taking LLaMA-13B as an example, it consumes around 1TB tokens with a total 59 MWh GPU power and emits around 23 tCO_{2eq} carbon. However, our model has consumed only 6.5 tCO_{2eq} carbon for 380B tokens, i.e., around 28.26% of the total carbon emission of the LLaMA-13B model. More training details and model implementation can be found in Section 3.4.

4.3 Natural language understanding

We evaluate the natural language understanding performance of OpenBA model on the SuperGLUE benchmark, which contains 13 sub-tasks. Since the BiFlan dataset contains partial training data of some testing tasks in SuperGLUE, we mainly compare OpenBA with models in the held-in setting (except GPT-3 [22]), i.e., these models have also been trained on the training data of some testing tasks in SuperGLUE. As we can observe in Table 6, the performance of OpenBA surpasses that of the BERT model [75] fine-tuned on the SuperGLUE training set and GPT-3, but is slightly behind that of the Flan-T5-XL [7] model.

We evaluate the reading comprehension ability of OpenBA with BELEBELE benchmark [71] and select the Chinese (Simplified), Chinese (Traditional), and English subsets for evaluation. We follow the official settings and compare with both LLMs and fine-tuned down-stream models, including Falcon [35], LLaMA [4,6], XLM-V [76], InfoXLM [77] and ChatGPT [33]. We provide all the instructions we use for zero-shot setting in the Supporting Information. As we can observe from Table 7, OpenBA can achieve outstanding results in the Chinese reading comprehension tasks, ranking just behind ChatGPT. For English reading comprehension tasks, the performance of OpenBA is comparable to that of the Falcon-40B model, which is trained with around 1TB tokens of multilingual data. It is also worth noting that OpenBA achieves better performance among multiple current open-source LLMs, including two strong LLaMA models and the Falcon-40B model, under the bilingual setting.

Table 6 Zero-shot results on SuperGLUE benchmark, where #Param. denotes the model parameters, Avg. denotes average accuracy, and Acc. indicates the model's predictive accuracy for the next word. The bold indicates the best performance and the underlined denotes the second-best performance.

Model metrics	#Param.	Avg.	BoolQ Acc.	CB Acc.	RTE Acc.	ReCoRD F1	ReCoRD EM	WSC Acc.
BERT-Large	340M	69.0	77.4	83.6	71.6	<u>72.0</u>	71.3	64.3
BERT-Large++	340M	71.5	79.0	<u>90.4</u>	79.0	<u>72.0</u>	<u>73.0</u>	64.3
Flan-T5-XL	3B	**79.3**	**89.3**	**91.2**	**90.4**	57.2	56.6	**84.9**
GPT3	175B	71.8	76.4	75.6	69.0	**91.1**	**90.0**	<u>80.1</u>
OpenBA	15B	<u>73.1</u>	<u>82.6</u>	85.6	<u>83.9</u>	69.4	68.8	76.0

Model metrics	#Param.	WiC Acc.	CoPA Acc.	MultiRC F1	MultiRC EM	AX_b MCC	AX_g GPS	AX_g Acc
BERT-Large	340M	**69.5**	70.6	70.0	24.0	23.0	<u>97.8</u>	51.7
BERT-Large++	340M	**69.5**	73.8	70.4	24.5	38.0	**99.4**	51.4
Flan-T5-XL	3B	65.7	**97.6**	**87.0**	**57.9**	**50.1**	97.2	**91.9**
GPT3	175B	49.4	<u>92.0</u>	75.4	30.5	21.1	90.4	55.3
OpenBA	15B	57.2	85.8	<u>77.1</u>	<u>38.9</u>	<u>40.8</u>	94.4	<u>70.2</u>

Table 7 Model performance on BELEBELE benchmark, where † denotes 5-shot setting, ‡ denotes full fine-tuning in English and ∗ denotes the zero-shot setting for instructed models. We report the accuracy score for all the models.

Model	#Param.	eng_Latn	zho_Hans	zho_Hant	Avg.
Falcon†	40B	77.2	66.0	62.2	68.5
LLaMA†	70B	<u>82.5</u>	64.6	57.7	68.2
InfoXLM‡	550M	79.3	74.6	72.4	75.4
XLM-V‡	1.2B	76.2	71.0	67.1	71.4
Baichuan2-Chat*	7B	65.6	62.2	58.1	62.0
LLaMA-2-Chat*	70B	78.8	62.4	59.3	66.8
GPT3.5-Turbo*	-	**87.7**	**77.6**	**76.3**	**80.5**
OpenBA*	15B	78.6	<u>75.2</u>	<u>73.7</u>	<u>75.8</u>

4.4 Natural language generation

We evaluate the natural language generation ability of our model on five tasks, including machine translation on the Flores [78] benchmark, text summarization on the CLTS benchmark [79], paraphrase task on the QQP dataset[8], text simplification on the WIKI-AUTO [80] dataset, and story generation on the ROC [81] dataset.

Summarization. To evaluate the summarization task under the held-out setting, we select a subset containing 100 sentences sampled from CLTS benchmark [79], which is excluded from the BiFlan dataset. Specifically, we prepend the task instruction before each test sentence (the task instruction is listed in the Supporting Information) and allow models to conduct zero-shot inference. We evaluate the generated results with Rouge-n metric [82] and report the results in Table 8. We observe that OpenBA can achieve the best performance on the Rouge-1 and Rouge-L scores, indicating that the content generated from OpenBA is faithful to the original text in the summarization task.

Machine translation. We compare the model performance on the bilingual machine translation tasks, including Chinese-to-English and English-to-Chinese translation, on the Flores [78] machine translation benchmark. We strictly follow the official settings by selecting 50 testing samples provided for each translation task. It is worth noting that all the models are under the held-out zero-shot setting. We report the BLUE-4 [83] scores in Table 9 and can observe that OpenBA can achieve the best performance on the Chinese-to-English translation task and obtain comparable results with the SOTA achieved by BatGPT on the English-to-Chinese translation task.

Text simplification and paraphrase. We evaluate the text simplification and paraphrase task of OpenBA on the WIKI AUTO and QQP datasets. We evaluate the model performance with BLUE, Distinct-n (D-n) metrics [84], Lexical Repetition (Rep-n, 4-gram repetition for n-times) [85], Mauve [86] and Semantic Similarity (SIM, semantic similarity between generations and corresponding prompts) [65] metrics, and report the model performance in Table 10. Based on the observation that OpenBA can

8) https://www.kaggle.com/c/quora-question-pairs.

Table 8 Model performance on CLTS subset containing 100 sentences sampled from CLTS test set. We report Rouge-1, Rouge-2 and Rouge-L score.

Model	#Param.	Rouge-1	Rouge-2	Rouge-L
ChatGLM	6B	<u>27.3</u>	**17.2**	<u>26.7</u>
Baichuan	7B	19.9	<u>14.4</u>	20.0
Baichuan2	7B	21.3	14.9	21.2
BatGPT	15B	25.6	12.2	25.0
OpenBA	15B	**30.2**	13.9	**28.6**

Table 9 Model performance on Flores subset containing 50 sentences sampled from Flores.

Model	#Param.	Zh⇒En	En⇒Zh	Avg.
ChatGLM	6B	17.2	32.5	24.9
Alpaca	7B	15.1	9.8	12.5
Baichuan2	7B	**23.5**	**40.2**	**31.9**
BatGPT	15B	23.1	<u>38.7</u>	<u>30.9</u>
OpenBA	15B	<u>23.3</u>	37.4	30.4

Table 10 Model performance on WIKU AUTO and QQP datasets.

Model	#Param.	WIKI AUTO					QQP				
		B-2 (↑)	D-2 (↑)	LR-2 (↓)	Mav (↑)	SIM (↑)	B-2 (↑)	D-2 (↑)	LR-2 (↓)	Mav (↑)	SIM (↑)
BatGPT	15B	25.5	<u>1.5</u>	89.9	96.5	<u>5.5</u>	19.4	1.6	67.6	58.3	7.6
ChatGLM	6B	**29.2**	1.4	<u>90.7</u>	<u>97.7</u>	4.0	**25.0**	<u>2.0</u>	63.9	<u>93.6</u>	5.6
MOSS	16B	27.8	<u>1.5</u>	82.9	96.8	5.4	19.3	1.4	<u>72.7</u>	37.2	<u>7.8</u>
Baichuan2	7B	23.1	1.3	92.0	95.4	5.0	23.3	1.9	55.5	93.1	6.1
OpenBA	15B	<u>27.9</u>	**1.9**	**75.6**	**99.1**	**6.6**	<u>22.7</u>	**2.0**	**48.0**	**94.4**	**7.9**

attain the best results on the Mav and SIM metrics, which evaluate semantic relevance with gold text and input text respectively, we can conclude that our model excels at capturing the overall semantic information of the input content and generating relevant content accordingly.

Story generation. We evaluate the open-domain generation capability of OpenBA on the ROC dataset, where the model should continue generating based on the existing context and the story plot. More concretely, we feed the model with the prompt directly and compare OpenBA with two other models: GPT-J [87] and OPT-13B [27], which are also trained on the Pile corpus. We randomly sample 100 generated cases and invite annotators to score the text from two aspects, including coherence between the generated text and the prompt, consistency of the generated text. The annotators are allowed to choose "Tie" if it is hard to distinguish two generation cases. As shown in Figure 4, we can observe our model can obtain strong performance on the coherence and consistency aspects.

4.5 Common sense reasoning

We evaluate the common sense reasoning ability of OpenBA on four benchmarks, including BBH (Table 11), MMLU (Table 12), CMMLU (Table 13), and C-Eval (Table 14). To ensure a fair comparison, we conduct all the evaluations under the held-out setting, follow the recommended setting of each benchmark, and compare with other strong LLMs under the same settings. For the MMLU and C-Eval benchmarks, we report the zero-shot, 5-shot, and 5-shot CoT results. For CMMLU, we report the zero-shot, 5-shot, and zero-shot CoT results. We also report the zero-shot performance on BBH benchmark. It is worth noting that the first block for each table is multilingual- or English-oriented models, the second block is Chinese-oriented models, and we rank the models in each block by model size. We can observe that, on all the benchmarks, OpenBA can achieve better performance than two strong Chinese-oriented models, i.e., ChatGLM [16] and BatGPT[9] [74], and obtain comparable results with Baichuan-7B model [10], which is trained on datasets much larger than ours, i.e., 1.2TB tokens. Furthermore, our model surpasses English-oriented models on most benchmarks and even outperforms some tasks where English-oriented models have over 100 billion parameters, e.g., BLOOM-176B, on the MMLU benchmark. Additionally, OpenBA can achieve comparable scores under both zero-shot and few-shot settings and even performs slightly

9) https://huggingface.co/MLP-lab/BatGPT-15B-sirius.

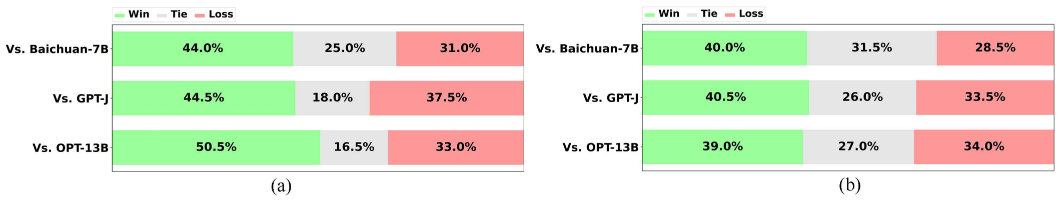


Figure 4 (Color online) Human evaluation results on the ROC dataset. (a) Coherence between the generated text and prompt; (b) consistency of the generated text.

Table 11 Model performance on the BBH benchmark. We report the accuracy score for all the models.

Model	#Param.	BBH
LLaMA	13B	**37.1**
ChatGLM	6B	31.3
Baichuan2	7B	41.6
BatGPT	15B	<u>34.1</u>
MOSS	16B	29.3
OpenBA	15B	<u>34.1</u>

Table 12 Model performance on MMLU benchmark, where #Param. denotes the model parameters, † denotes 5-shot, ‡ denotes 0-shot, and ∗ represents the chain-of-thought.

Model	#Param.	Humanities	STEM	Social sciences	Other	Average
LLaMA2†	7B	43.4	36.9	51.7	52.6	46.0
LLaMA2†	13B	**45.0**	<u>35.8</u>	**53.8**	**53.3**	**46.9**
BLOOM†	176B	34.1	**36.8**	41.5	46.5	39.1
ChatGLM†	6B	35.4	31.3	41.0	40.5	36.9
Baichuan2†	7B	51.2	44.6	61.5	60.5	54.2
BatGPT†	15B	35.4	33.5	36.3	37.0	36.7
MOSS†	16B	30.5	29.3	33.8	34.4	31.9
OpenBA†	15B	34.6	29.8	40.1	40.0	36.0
OpenBA‡	15B	<u>38.7</u>	33.8	45.0	43.6	40.2
OpenBA∗	15B	36.7	31.4	42.8	42.3	38.2

Table 13 Performance on CMMLU benchmark, where #Param. denotes the model parameters, and ∗ denotes chain-of-thought. We report the 5-shot and 0-shot performance with diagonal bar division.

Model	#Param.	Humanities	STEM	Social science	Other	China-specific	Average
Falcon	40B	<u>43.5</u>/41.3	33.3/31.1	44.3/40.9	<u>44.8</u>/40.6	<u>39.5</u>/36.1	<u>41.5</u>/38.5
LLaMA	65B	40.2/34.5	34.5/31.1	41.6/36.1	42.9/37.9	37.0/32.9	39.8/34.9
ChatGLM	6B	39.2/42.9	32.4/32.2	39.7/44.8	38.6/42.6	37.7/41.9	37.5/40.8
Baichuan2	7B	61.0/59.2	43.6/42.3	61.8/59.9	60.8/58.7	58.3/56.6	56.8/55.0
BatGPT	15B	35.5/36.5	<u>35.0</u>/33.7	36.3/38.1	42.1/46.9	37.9/38.3	37.2/38.5
OpenBA	15B	40.9/40.9	33.5/33.8	<u>45.2</u>/44.7	44.5/43.6	39.1/38.6	<u>41.5</u>/41.2
OpenBA∗	15B	30.0	37.6	40.6	39.2	36.4	37.0

Table 14 Model performance on C-Eval benchmark, where ∗ denotes chain-of-thought and Avg. represents average accuracy.

Model	#Param.	STEM	Social science	Humanities	Others	Avg.	Avg. (Hard)
LLaMA	65B	<u>37.8</u>	45.6	36.1	37.1	38.8	<u>31.7</u>
ChatGLM	6B	33.3	48.3	41.3	38.0	38.9	29.2
Baichuan2	7B	47.8	67.5	58.7	53.1	55.0	36.8
MOSS-moon-sft	16B	31.6	37.0	33.4	32.1	33.1	28.4
GLM-130B	130B	36.7	**55.8**	**47.7**	**43.0**	**44.0**	30.7
OpenBA	15B	34.8	46.6	41.1	<u>41.5</u>	39.8	31.1
OpenBA∗	15B	30.7	43.7	40.9	35.2	36.3	27.0

better under the zero-shot setting, indicating that the OpenBA model has a strong instruction-following capability.

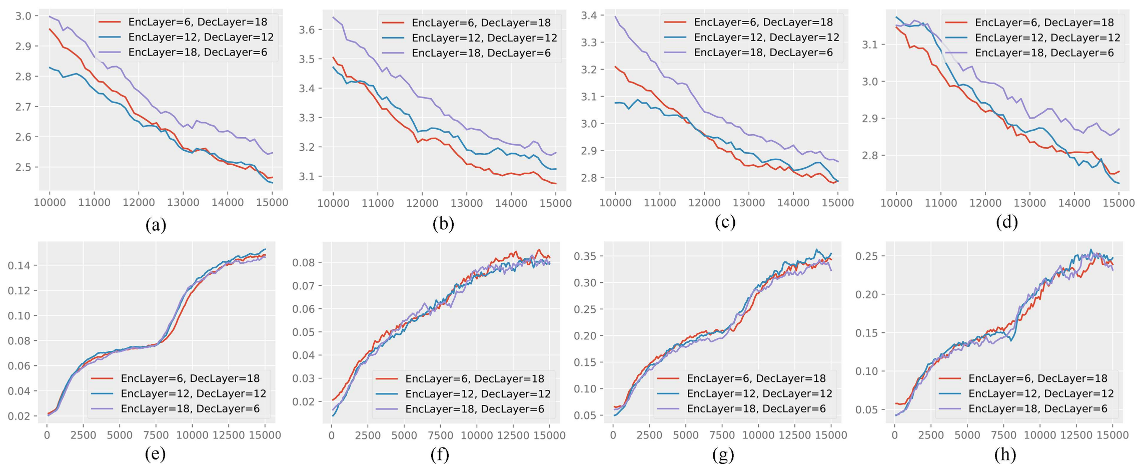


Figure 5 (Color online) The performance in terms of loss and accuracy of the three model configurations across four denoising tasks. The first row of figures illustrates the loss performance ((a)–(d)), while the second row depicts the accuracy ((e)–(h)). The four columns represent the four tasks: R-denoising, S-denoising, X-denoising, and a combination of the three, respectively.

5 Analysis

5.1 OpenBA model architecture selection

Our asymmetric shallow-encoder deep-decoder model architecture stems from the following considerations.

• **Enhanced generative capabilities.** For the three tasks in UL2, i.e., R-denoising, S-denoising, and X-denoising, a deeper decoder setup is particularly effective for the S-denoising task, which reflects the model's language modeling ability.

• **Potential acceleration in dialogue inference.** Decoder-only architectures similar to GPT have already achieved excellent results in multi-turn dialogue tasks. However, for encoder-decoder models, how to store dialogue history presents a significant challenge. A common approach is to embed the dialogue history into the encoder's input. However, continuously altering this history results in increased computational costs in the encoder, and it's not amenable to acceleration via KV-caching. To address this challenge, we can place the dialogue history into the decoder. This shift imposes a greater demand on the decoder's capabilities. Thus, we explore training a deeper decoder to endow it with enhanced capabilities.

We conduct experiments to explore the influence of the model architecture, where we train the model with the UL2 training objective. Specifically, we set the batch size as 128 and the sequence length as 570/380. We validate the model performance after 15k training steps.

Model configuration. We mainly explore three model structures: (1) a shallow encoder with a deep decoder, (2) a deep encoder with a shallow decoder, and (3) the encoder and decoder with equal depth. We assess their performance metrics across the R-denoising, S-denoising, and X-denoising tasks to learn their respective merits. To maintain consistent parameter counts across different configurations, we adopt these layer structures: (1) EncoderLayer=18, DecoderLayer=6, (2) EncoderLayer=6, DecoderLayer=18, and (3) EncoderLayer=DecoderLayer=12.

Evaluation metric. To get a direct view of the model performance pre-trained from scratch, we choose Loss and Acc. as convenient metrics. Specifically, we construct validation sets for R-denoising, S-denoising, X-denoising, and a combination of the three and test the model's performance throughout the training process. Acc. indicates the model's predictive accuracy for the next word:

$$\text{Acc.} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(\text{argmax}_{w\in V}P(\boldsymbol{x}_i = w|\boldsymbol{x}_{<i},\theta) = \boldsymbol{x}_i), \tag{3}$$

where n denotes the sequence length, V denotes the vocabulary size and $\mathbb{I}$ is an indicator function.

Analysis. Figure 5 shows our results. We can conclude the following.

• As a measurement of the model's generation ability, the S-denoising task is generally more challenging to learn. This is evident as, regardless of the model configuration, the S-denoising task consistently has

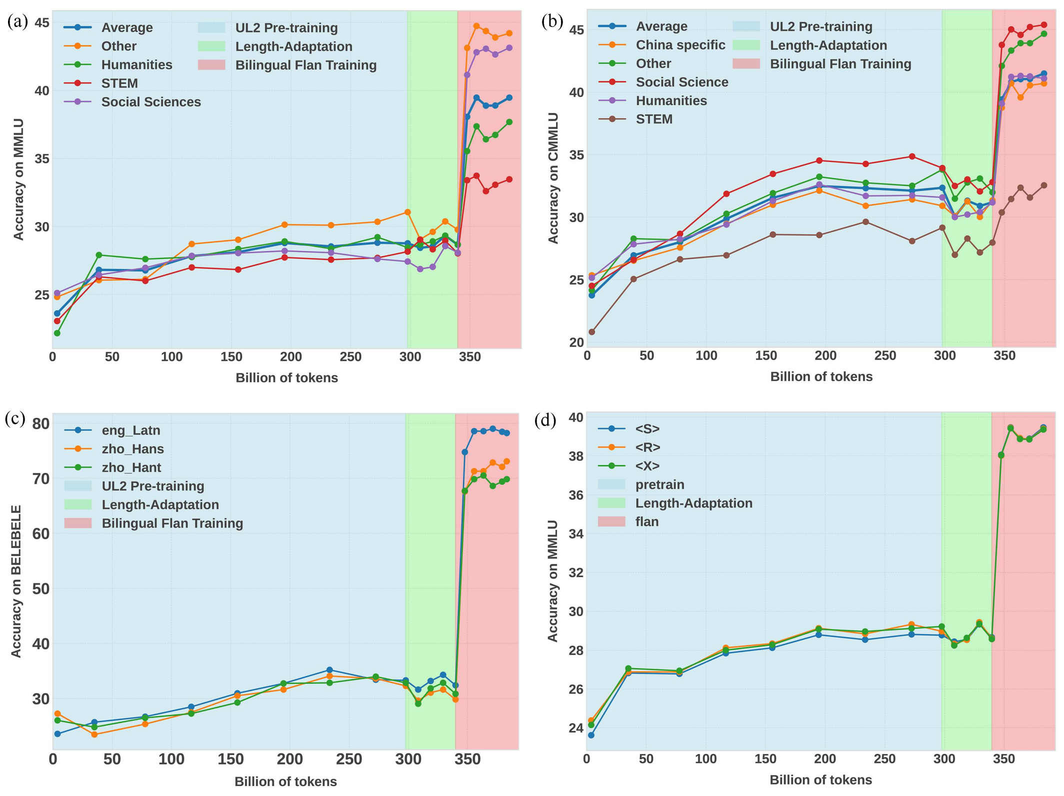


Figure 6 (Color online) Evolution of model performance during training. (a) Model performance evolution on the MMLU benchmark; (b) model performance evolution on the CMMLU benchmark; (c) model performance evolution on the BELEBELE benchmark; (d) the influence of extra paradigm tokens along with pre-training.

a higher loss and a lower accuracy.

- The model with a shallow encoder and deep decoder configuration performs better on the S-denoising task (from Figures 5(b) and (f)), though it doesn't outperform the balanced setup across all three tasks (from Figures 5(b) and (f)).

5.2 Evolution of performance during training

In this section, we evaluate the performance of OpenBA at various stages of the overall training. We employ three benchmarks for evaluation, including MMLU for English common sense reasoning, CMMLU for Chinese common sense reasoning, and BELEBELE for reading comprehension. As shown in Figures 6(a)–(c), the performance on most tasks increases with the number of training steps during the UL2 pre-training stage, experiences slight fluctuations during the length-adaptation stage, and exhibits a significant improvement during the bilingual Flan training stage. The emergence curves of Chinese and English are similar, indicating that our bilingual Flan dataset effectively enhances multi-language task performance on held-out tasks. Moreover, we measure the performance on MMLU when given different extra paradigm tokens, i.e., {`<R>`, `<S>`, `<X>`}. We find that the performance with different extra paradigm tokens shows differences during the UL2 pre-training stage, while these differences gradually diminish in the subsequent stages. This might be attributed to the fact that we utilize these extra paradigm tokens to guide the mode-switching only in the first stage for different UL2 tasks. Specifically, the performance for the S-denoising task in continuous writing is slightly inferior compared to the X-denoising and R-denoising tasks for masked span recovery.

5.3 Suggestions for model architecture selection

It is worth noting that no one-size-fits-all model architecture can adapt to all tasks. However, setting aside factors like the quality of data and training strategies, which can significantly impact final performance, we have outlined the following suggestions for model architecture selection based on the aforementioned experimental results.

• **Encoder-only architecture**. Encoder-only model, e.g., BERT, is competent for understanding tasks. For instance, in the SuperGLUE benchmark (Table 6), the BERT-Large model, though with only 340M parameters, can achieve comparable results with GPT3 with 175B parameters. This benefit arises from the encoder-only model's bi-directional attention and mask-filling training paradigm.

• **Decoder-only architecture**. If the task is centered around text generation, such as story generation and creative writing, decoder-only models often provide strong performance. These models are optimized for producing coherent and contextually rich text sequences, making them a go-to choice for applications where the primary goal is to generate text.

• **Encoder-decoder architecture**. Encoder-decoder model excels in scenarios that require both understanding and generation, such as tasks like summarization and text simplification. These tasks necessitate a deep comprehension of the original text's semantics as well as the capability to generate corresponding content. Moreover, in situations where there's a discrepancy between the distribution of the input and output, e.g., machine translation and multi-modality, the encoder-decoder model is adept at handling these tasks. However, it's important to note that the encoder-decoder model is not the best choice for modeling long sequences due to: (1) the encoder and the decoder need to do their own work of encoding, which presents substantial challenges in determining their optimal longest sequence lengths, and (2) the encoder adds extra memory usage. This means that, compared to models with only a decoder, encoder-decoder models will use more of the GPU's memory, which could be a problem for long-sequence modeling tasks.

5.4 Training efficiency

In this section, we analyze why OpenBA requires fewer GPU hours. Beyond the factor of using less training data and the inherent performance of the machine itself, such as disk I/O operations and network communication, our discussion primarily focuses on aspects related to data handling and training methodologies.

Data efficiency. Experimental results indicate that there is a significant correlation between model performance and data quality. As illustrated in Figure 6, there is a marked improvement in the model's performance when the Bi-Flan dataset is utilized. This suggests that we can achieve decent model performance with a limited amount of GPU time. The efficiency in model training can be partially attributed to the high-quality dataset employed.

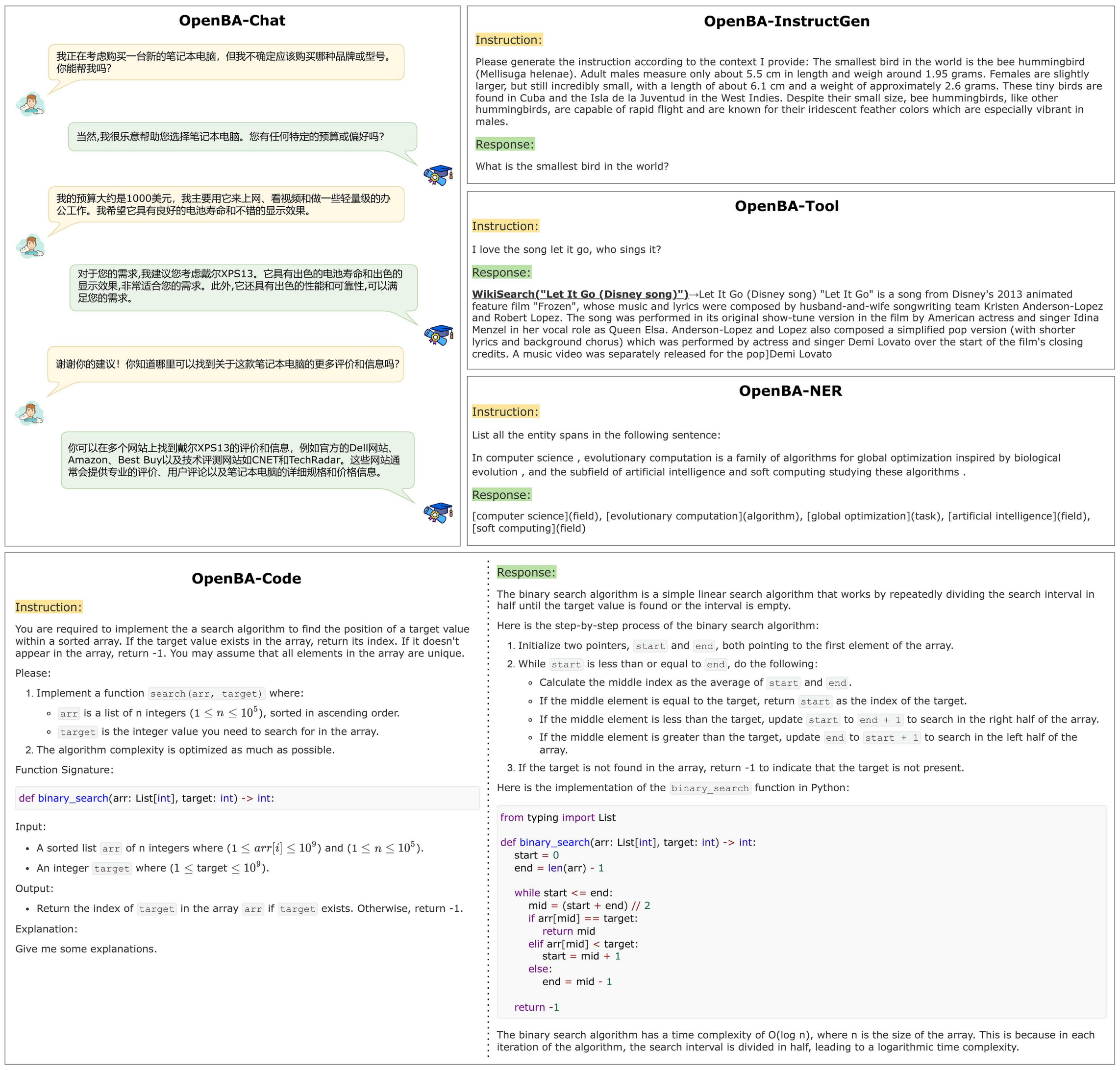
Framework efficiency. In Sections 3.4, we have summarized the techniques we employed, which significantly contributed to our training efficiency. Notably, we re-implement the Megatron-LM library, incorporating the approaches detailed in Section 3.4 to accelerate the training process.

Explosion loss skiping. Throughout the training of LLMs, episodes of training instability often occur [27, 88]. One plausible cause for this is the ingestion of data samples that deviate from the dataset distribution or contain anomalous elements, such as hyperlinks, mathematical symbols, or corrupt characters. To avoid such an issue, we adopt the explosion loss skiping strategy, whereby if a calculated loss exceeds a certain threshold substantially large figure (10000, for instance) will discard the gradients for that step and proceed immediately to the following step. This method is incredibly effective in preventing the unnecessary expense of training costs due to rollback checkpoints and re-warmup training procedures.

6 OpenBA-X: downstream task adaptation

As indicated by current works [89], traditional benchmarks, such as MMLU and HELM, primarily assessing core capabilities on a limited set of tasks, often fail to reflect a model's performance in more open-ended, real-world interactions[10)]. After Stage III, we conduct supervised fine-tuning for OpenBA on four downstream tasks to reveal the potential of down-stream application of the OpenBA model, including bilingual multi-turn dialogue (OpenBA-Chat), code generation (OpenBA-Code), instruction generation (OpenBA-InstructGen), and tool retrieval (OpenBA-Tool). Figure 7 provides a few representative examples on different downstream tasks. In Subsections 6.1 to 6.4, we will provide details about the collection and processing of the downstream datasets. It is worth mentioning that we use the S-denoising

10) It is worth noting that traditional benchmarks do not accurately reflect a model's performance in real-world applications. Evaluating the capabilities of LLMs on real-world tasks is both costly (requiring human evaluation) and potentially biased (when using LLMs as judges). We urge readers to carefully consider these limitations and exercise caution when using our model in practical scenarios.



OpenBA-Chat

我正在考虑购买一台新的笔记本电脑，但我不确定应该购买哪种品牌或型号。你能帮我吗?

当然,我很乐意帮助您选择笔记本电脑。您有任何特定的预算或偏好吗?

我的预算大约是1000美元，我主要用它来上网、看视频和做一些轻量级的办公工作。我希望它具有良好的电池寿命和不错的显示效果。

对于您的需求,我建议您考虑戴尔XPS13。它具有出色的电池寿命和出色的显示效果,非常适合您的需求。此外,它还具有出色的性能和可靠性,可以满足您的需求。

谢谢你的建议！你知道哪里可以找到关于这款笔记本电脑的更多评价和信息吗?

你可以在多个网站上找到戴尔XPS13的评价和信息，例如官方的Dell网站、Amazon、Best Buy以及技术评测网站如CNET和TechRadar。这些网站通常会提供专业的评价、用户评论以及笔记本电脑的详细规格和价格信息。

OpenBA-InstructGen

Instruction:

Please generate the instruction according to the context I provide: The smallest bird in the world is the bee hummingbird (Mellisuga helenae). Adult males measure only about 5.5 cm in length and weigh around 1.95 grams. Females are slightly larger, but still incredibly small, with a length of about 6.1 cm and a weight of approximately 2.6 grams. These tiny birds are found in Cuba and the Isla de la Juventud in the West Indies. Despite their small size, bee hummingbirds, like other hummingbirds, are capable of rapid flight and are known for their iridescent feather colors which are especially vibrant in males.

Response:

What is the smallest bird in the world?

OpenBA-Tool

Instruction:

I love the song let it go, who sings it?

Response:

WikiSearch("Let It Go (Disney song)")→Let It Go (Disney song) "Let It Go" is a song from Disney's 2013 animated feature film "Frozen", whose music and lyrics were composed by husband-and-wife songwriting team Kristen Anderson-Lopez and Robert Lopez. The song was performed in its original show-tune version in the film by American actress and singer Idina Menzel in her vocal role as Queen Elsa. Anderson-Lopez and Lopez also composed a simplified pop version (with shorter lyrics and background chorus) which was performed by actress and singer Demi Lovato over the start of the film's closing credits. A music video was separately released for the pop]Demi Lovato

OpenBA-NER

Instruction:

List all the entity spans in the following sentence:

In computer science , evolutionary computation is a family of algorithms for global optimization inspired by biological evolution , and the subfield of artificial intelligence and soft computing studying these algorithms .

Response:

[computer science](field), [evolutionary computation](algorithm), [global optimization](task), [artificial intelligence](field), [soft computing](field)

OpenBA-Code

Instruction:

You are required to implement the a search algorithm to find the position of a target value within a sorted array. If the target value exists in the array, return its index. If it doesn't appear in the array, return -1. You may assume that all elements in the array are unique.

Please:

1. Implement a function `search(arr, target)` where:
   - `arr` is a list of n integers ($1 \leq n \leq 10^5$), sorted in ascending order.
   - `target` is the integer value you need to search for in the array.
2. The algorithm complexity is optimized as much as possible.

Function Signature:

```
def binary_search(arr: List[int], target: int) -> int:
```

Input:

- A sorted list `arr` of n integers where ($1 \leq arr[i] \leq 10^9$) and ($1 \leq n \leq 10^5$).
- An integer `target` where ($1 \leq$ target $\leq 10^9$).

Output:

- Return the index of `target` in the array `arr` if `target` exists. Otherwise, return -1.

Explanation:

Give me some explanations.

Response:

The binary search algorithm is a simple linear search algorithm that works by repeatedly dividing the search interval in half until the target value is found or the interval is empty.

Here is the step-by-step process of the binary search algorithm:

1. Initialize two pointers, `start` and `end`, both pointing to the first element of the array.
2. While `start` is less than or equal to `end`, do the following:
   - Calculate the middle index as the average of `start` and `end`.
   - If the middle element is equal to the target, return `start` as the index of the target.
   - If the middle element is less than the target, update `start` to `end + 1` to search in the right half of the array.
   - If the middle element is greater than the target, update `end` to `start + 1` to search in the left half of the array.
3. If the target is not found in the array, return -1 to indicate that the target is not present.

Here is the implementation of the `binary_search` function in Python:

```
from typing import List

def binary_search(arr: List[int], target: int) -> int:
    start = 0
    end = len(arr) - 1

    while start <= end:
        mid = (start + end) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            start = mid + 1
        else:
            end = mid - 1

    return -1
```

The binary search algorithm has a time complexity of O(log n), where n is the size of the array. This is because in each iteration of the algorithm, the search interval is divided in half, leading to a logarithmic time complexity.

Figure 7 (Color online) Examples of OpenBA-X model on different downstream tasks. For the OpenBA-Chat model, we show the Chinese dialogue results. It is worth noting that there may be unrealistic content due to model hallucinations [90].

strategy for fine-tuning all downstream tasks, i.e., adding the “<S>” token before each target text that is fed to the decoder. We list all the instruction templates in the Supporting Information.

6.1 OpenBA-Chat: bilingual multi-turn dialogue

Dataset collection. We build bilingual supervised multi-turn dialogue data from three distinct sources: DialogStudio [91], BELLE [92], and ShareGPT[11]. We use the DialogStudio dataset for English dialogue data as it contains diverse conversations for various scenarios. As for Chinese dialogue data, we employ the BELLE dataset and ShareGPT data processed by others[12]. We filter out the overly simple conversations based on their length, as well as the content containing model identity information, e.g., “I am ChatGPT”. More importantly, we manually annotate 40 bilingual conversations to identify OpenBA and repeat them ten times before adding them to the training dataset.

Dataset processing. Given T turns of conversations involving two actors H and A in a dialogue, the data can be written as: $\boldsymbol{S} = (H_1, A_1, H_1, A_1, \cdots, H_t, A_t, \cdots, H_T, A_T)$, where (H_t, A_t) represents the t-th turn of the conversation. In order to enable the model to perceive the dialogue history and respond

11) https://huggingface.co/datasets/RyokoAI/ShareGPT52K.
12) https://github.com/PhoebusSi/Alpaca-CoT.

based on historical information, we process each dialogue data $\boldsymbol{S}$ into the set $\boldsymbol{D}$:

$$\boldsymbol{D} = \bigcup_{t=1}^{T} \{\mathrm{Input}_t, \mathrm{Target}_t\} = \bigcup_{t=1}^{T} \{(H_1, A_1, H_2, A_2, \cdots, H_t), (A_t)\},$$

where Input_t represents the input sequence, and Output_t represents the response sequence. The template to create the conversation for each instance is shown below:

Input: "Human: $\{H_0\}$ Assistant: $\{A_0\}$ $\cdots$ Human: $\{H_t\}$ Assistant:"
Output: "$\{A_t\}$"

6.2 OpenBA-Code: code generation

Dataset collection. For code generation, we mainly focus on the Python language. We choose a filtered version of the Evol-Instruct dataset [93], containing 26588 code samples13).

Dataset processing. The original tokenizer of OpenBA would ignore consecutive spaces, thereby erasing the indentation information within the code. To tackle this issue, we incorporate three special tokens into the vocabulary: the tab character '\t', the newline character '\n', and consecutive spaces. We directly utilize the instructions from the original dataset as the instructions vary for different code contents.

6.3 OpenBA-InstructGen: instruction generation

Dataset collection. We construct a bilingual dataset for the instruction generation task by reversing the original instruction dataset [52, 94]. Specifically, we utilize the DollyV2 dataset [95], Lima [96] and its corresponding Chinese version Lima-Chinese14). More concretely, we repeat the Chinese corpus twice and combine them with the English dataset for language balance.

Dataset processing. Given an instruction "Instruction" and its corresponding answer "Answer", we utilize the following templates (including English and Chinese) to wrap each pair.

Input: Please generate the instruction according to the text I provide: {Answer}.
Output: {Instruction}.
Input: 请你根据提供的文本生成对应的指令: {Answer}
Output: {Instruction}

6.4 OpenBA-tool: Tool retrieval

Dataset collection. In order to enable the OpenBA model to respond to user instructions with the help of external tools [97, 98], we select Toolformer-Retrieval dataset15), which is designed for retrieval task. For each instance, it is presented in the following format.

WikiSearch({Query Input}) $\rightarrow$ {Recalled Results}, where "WikiSearch(" denotes the beginning of calling external tool (Wikipedia here), "{Query Input}" is the generated query input for the tool, and "{Recalled Results}" represents the results returned by invoking the tool.

Dataset processing. We utilize the instructions provided by the Toolformer-Retrieval dataset directly and discard the cases that fail to call tools. For simplicity, we utilize the model's output as a proxy for the actual retrieval results. One can employ the results generated by the model to invoke downstream tools.

6.5 OpenBA-NER: named entity recognition

Dataset collection. We employ the Pile-NER dataset [99] for training, which contains around 240000 entities spanning 13000 unique entity categories. Subsequently, we assess our model, OpenBA-NER, using the MIT [100] and CrossNER [101] datasets, where the entity labels are largely unseen during the training phase.

Dataset processing. Given a sentence X and its corresponding answer Y, our prompt is constructed as follows:

Input: List all the entity in the sentence. Use the specific entity tags: {labels}. Sentence: {sentence}

13) https://huggingface.co/datasets/mlabonne/Evol-Instruct-Python-26k.
14) https://huggingface.co/datasets/paralym/lima-chinese.
15) https://huggingface.co/datasets/kentsui/open-toolformer-retrieval.

Table 15 Out-of-domain evaluation performance on MIT [101] and CrossNER [100] benchmark.

Model	Movie	Restaurant	AI	Literature	Music	Politics	Science	Avg.
UniNER-7B	42.4	31.7	53.5	59.4	65.0	60.8	61.1	53.4
UniNER-13B	48.7	36.2	54.2	60.9	64.5	61.4	63.5	55.6
InstructUIE-11B	63.0	21.0	49.0	47.2	53.2	48.2	49.3	47.3
OpenBA-NER-15B	51.4	37.4	58.5	54.6	69.1	68.4	68.2	58.2

Output: [Entity$_1$](Type$_1$), [Entity$_2$](Type$_2$), ...

Out-of-domain evaluation. To evaluate our model's performance on out-of-domain datasets, we present the strict entity F_1 score on the aforementioned evaluation datasets in Table 15. We also incorporate two strong baselines for comparison, i.e., UniversalNER [99], based on LLaMA [4], and InstructUIE [102], based Flan-T5-xxl (11B) [7]. We also include comparisons with two strong baselines: (1) UniversalNER, which utilizes LLaMA as backbone models, and (2) InstructUIE, which is fine-tuned on a broad range of IE datasets based on Flan-T5-xxl (11B) [7]. The results demonstrate the significant superiority of our model compared to the others.

7 Conclusion and future work

In this report, we present OpenBA, an open-sourced 15B bilingual asymmetric Seq2Seq model pre-trained from scratch. We provide all the necessary details to pre-train an asymmetric Seq2Seq model from scratch, including (1) how to construct and process the pre-training data, (2) how to construct the bilingual Flan data collection, (3) the implementation details of model architectures, configurations, objectives, and training pipelines. We also release our codes to supplement the descriptions of this report. On a variety of benchmarks, though fed with 380B tokens, OpenBA obtains remarkable performance, e.g., CMMLU and BELEBELE, and even surpasses the models consuming significantly more data. In addition, we also provide the details of fine-tuning OpenBA on five different downstream tasks and release the checkpoints of these models for quick implementation. However, due to the limitation of computational resources, OpenBA-15B is trained on a corpus of 380 billion tokens, which is considerably less compared to the prevalent strong LLMs like LLaMA, which utilizes 2 TB of data for training. In addition, the 15 billion model parameter presents challenges in both training efficiency and real-world applications. In the future, we plan to employ model compression techniques to condense the OpenBA-15B model into smaller variants, on which we will continue advanced pre-training to strengthen their performance.

Acknowledgements This work was supported by National Natural Key R&D Program of China (Grant No. 2020AAA0108604), National Natural Science Foundation of China (Grant Nos. 62206194, 62106165), and Natural Science Foundation of Jiangsu Province, China (Grant No. BK20220488). We sincerely thank the GPU sponsor of the Supercomputing Center in Yancheng and technical advice from Bowen YAN and Jianye HOU.

Supporting information Appendixes A and B. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models. 2020. ArXiv:2001.08361
2 Clark A, De Las Casas D, Guy A, et al. Unified scaling laws for routed language models. In: Proceedings of International Conference on Machine Learning. PMLR, 2022. 4057–4086
3 Hoffmann J, Borgeaud S, Mensch A, et al. Training compute-optimal large language models. 2022. ArXiv:2203.15556
4 Touvron H, Lavril T, Izacard G, et al. LLaMA: open and efficient foundation language models. 2023. ArXiv:2302.13971
5 Scao T L, Fan A, Akiki C, et al. Bloom: a 176B-parameter open-access multilingual language model. 2022. ArXiv:2211.05100
6 Touvron H, Martin L, Stone K, et al. LLaMA 2: open foundation and fine-tuned chat models. 2023. ArXiv:2307.09288
7 Chung H W, Hou L, Longpre S, et al. Scaling instruction-finetuned language models. 2022. ArXiv:2210.11416
8 Soltan S, Ananthakrishnan S, FitzGerald J, et al. Alexatm 20B: few-shot learning using a large-scale multilingual Seq2Seq model. 2022. ArXiv:2208.01448
9 Zeng A, Liu X, Du Z, et al. GLM-130B: an open bilingual pre-trained model. 2022. ArXiv:2210.02414
10 Inc. B. Baichuan-7b. https://github.com/baichuan-inc/Baichuan-7B, 2023
11 Wang H, Liu C, Xi N, et al. Huatuo: tuning LLaMA model with chinese medical knowledge. 2023. ArXiv:2304.06975
12 Leng Z, Chen Q, Li C. Luotuo: an instruction-following chinese language model, lora tuning on LLaMA. https://github.com/LC1332/Chinese-alpaca-lora, 2023
13 Chen Z, Jiang F, Chen J, et al. Phoenix: democratizing ChatGPT across languages. 2023. ArXiv:2304.10453

14 Cui Y, Yang Z, Yao X. Efficient and effective text encoding for chinese LLaMA and alpaca. 2023. ArXiv:2304.08177
15 Sun T X, Zhang X T, He Z F, et al. MOSS: an open conversational large language model. Mach Intell Res, 2024, 21: 888–905
16 Du Z, Qian Y, Liu X, et al. GLM: general language model pretraining with autoregressive blank infilling. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022. 320–335
17 Longpre S, Hou L, Vu T, et al. The flan collection: designing data and methods for effective instruction tuning. 2023. ArXiv:2301.13688
18 Tay Y, Dehghani M, Tran V Q, et al. UL2: unifying language learning paradigms. In: Proceedings of the Eleventh International Conference on Learning Representations. Kigali, 2022
19 Zheng L, Chiang W L, Sheng Y, et al. Judging LLM-as-a-judge with mT-bench and chatbot arena, In: Proceedings of the 37th International Conference on Neural Information Processing Systems. New Orleans, 2023. 46595–46623
20 Schaller R R. Moore's law: past, present and future. IEEE Spectr, 1997, 34: 52–59
21 Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics. New Orleans, 2018
22 Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. In: Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Vancouver, 2020. 1877–1901
23 Zeng W, Ren X, Su T, et al. Pangu-*alpha*: large-scale autoregressive pretrained chinese language models with auto-parallel computation. 2021. ArXiv:2104.12369
24 Sun Y, Wang S, Feng S, et al. ERNIE 3.0: large-scale knowledge enhanced pre-training for language understanding and generation. 2021. ArXiv:2107.02137
25 Zhang M, Li J. A commentary of GPT-3 in MIT technology review 2021. Fundam Research, 2021, 1: 831–833
26 Zhang Z, Gu Y, Han X, et al. CPM-2: large-scale cost-effective pre-trained language models. AI Open, 2021, 2: 216–224
27 Zhang S, Roller S, Goyal N, et al. OPT: open pre-trained transformer language models. 2022. ArXiv:2205.01068
28 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). California, 2017. 6000–6010
29 Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners. OpenAI blog, 2019, 1: 9
30 Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res, 2020, 21: 5485–5551
31 Rae J W, Borgeaud S, Cai T, et al. Scaling language models: methods, analysis & insights from training gopher. 2021. ArXiv:2112.11446
32 Smith S, Patwary M, Norick B, et al. Using deepspeed and megatron to train megatron-turing NLG 530B, a large-scale generative language model. 2022. ArXiv:2201.11990
33 Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans, 2022. 27730–27744
34 Bubeck S, Chandrasekaran V, Eldan R, et al. Sparks of artificial general intelligence: early experiments with GPT-4. 2023. ArXiv:2303.12712
35 Penedo G, Malartic Q, Hesslow D, et al. The refinedweb dataset for falcon LLM: outperforming curated corpora with web data, and web data only. 2023. ArXiv:2306.01116
36 Pan X, Zhang M, Ji S, et al. Privacy risks of general-purpose language models. In: Proceedings of 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020. 1314–1331
37 Chowdhery A, Narang S, Devlin J, et al. PmLM: scaling language modeling with pathways. 2022. ArXiv:2204.02311
38 Hendrycks D, Burns C, Basart S, et al. Measuring massive multitask language understanding. 2020. ArXiv:2009.03300
39 Li H, Zhang Y, Koto F, et al. CMMLU: measuring massive multitask language understanding in chinese. 2023. ArXiv:2306.09212
40 Fu Z, Lam W, Yu Q, et al. Decoder-only or encoder-decoder? Interpreting language model as a regularized encoder-decoder. 2023. ArXiv:2304.04052
41 Wang A, Pruksachatkun Y, Nangia N, et al. SuperGLUE: a stickier benchmark for general-purpose language understanding systems. In: Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). 2019. 3266–3280
42 Huang Y, Bai Y, Zhu Z, et al. C-Eval: a multi-level multi-discipline chinese evaluation suite for foundation models. 2023. ArXiv:2305.08322
43 Zhang S, Dong L, Li X, et al. Instruction tuning for large language models: a survey. 2023. ArXiv:2308.10792
44 Wei J, Bosma M, Zhao V Y, et al. Finetuned language models are zero-shot learners. 2021. ArXiv:2109.01652
45 Sanh V, Webson A, Raffel C, et al. Multitask prompted training enables zero-shot task generalization. 2021. ArXiv:2110.08207
46 Nye M, Andreassen A J, Gur-Ari G, et al. Show your work: scratchpads for intermediate computation with language models. 2021. ArXiv:2112.00114
47 Wei J, Wang X, Schuurmans D, et al. Chain of thought prompting elicits reasoning in large language models. 2022. ArXiv:2201.11903
48 Wang X, Wei J, Schuurmans D, et al. Self-consistency improves chain of thought reasoning in language models. 2022. ArXiv:2203.11171
49 Zelikman E, Wu Y, Mu J, et al. STaR: bootstrapping reasoning with reasoning. In: Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). New Orleans, 2022. 15476–15488
50 Wu Y, Zhao Y, Li Z, et al. Improving cross-task generalization with step-by-step instructions. 2023. ArXiv:2305.04429
51 Xu C, Sun Q, Zheng K, et al. WizardLM: empowering large language models to follow complex instructions. 2023. ArXiv:2304.12244
52 Taori R, Gulrajani I, Zhang T, et al. Stanford alpaca: an instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023

53 Conover M, Hayes M, Mathur A, et al. Free dolly: introducing the world first truly open instruction-tuned LLM. Company Blog, 2023

54 Chaudhary S. Code Alpaca: an instruction-following LLaMA model for code generation. https://github.com/sahil280114/codealpaca, 2023

55 Zhang Z, Zhang A, Li M, et al. Automatic chain of thought prompting in large language models. In: Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023). 2023

56 Gao L, Biderman S, Black S, et al. The Pile: an 800GB dataset of diverse text for language modeling. 2020. ArXiv:2101.00027

57 Ding M, Yang Z, Hong W, et al. CogView: mastering text-to-image generation via transformers. In: Proceedings of the 35th International Conference on Neural Information Processing System. Red Hook, 2021. 19822–19835

58 Zhang B, Sennrich R. Root mean square layer normalization. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Red Hook, 2019. 12381–12392

59 Su J, Lu Y, Pan S, et al. RoFormer: enhanced transformer with rotary position embedding. 2021. ArXiv:2104.09864

60 Shazeer N. GLU variants improve transformer. 2020. ArXiv:2002.05202

61 Xue L, Constant N, Roberts A, et al. mT5: a massively multilingual pre-trained text-to-text transformer. 2020. ArXiv:2010.11934

62 Barshan E, Fieguth P. Stage-wise training: an improved feature learning strategy for deep models. In: Proceedings of Feature Extraction: Modern Questions and Challenges at NIPS 2015. PMLR, 2015. 49–59

63 Lewis M, Liu Y, Goyal N, et al. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2019, 7871–7880

64 Min S, Lewis M, Zettlemoyer L, et al. MetaICL: learning to learn in context. 2021. ArXiv:2110.15943

65 Guan J, Mao X, Fan C, et al. Long text generation by modeling sentence-level and discourse-level coherence. 2021. ArXiv:2105.08963

66 Grun P. Introduction to infiniband for end users. White paper, InfiniBand Trade Association, 2010, 55

67 Shoeybi M, Patwary M, Puri R, et al. Megatron-LM: training multi-billion parameter language models using model parallelism. 2019. ArXiv:1909.08053

68 Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: memory optimizations toward training trillion parameter models. In: Proceedings of SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020. 1–16

69 Nijkamp E, Pang B, Hayashi H, et al. CodeGen: an open large language model for code with multi-turn program synthesis. 2022. ArXiv:2203.13474

70 Zhou K, Zhu Y, Chen Z, et al. Don't make your LLM an evaluation benchmark cheater. 2023. ArXiv:2311.01964

71 Bandarkar L, Liang D, Muller B, et al. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. 2023. ArXiv:2308.16884

72 Suzgun M, Scales N, Schärli N, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. 2022. ArXiv:2210.09261

73 Wu C J, Raghavendra R, Gupta U, et al. Sustainable AI: environmental implications, challenges and opportunities. In: Proceedings of Machine Learning and Systems. 2022. 795–813

74 Li Z, Zhang S, Zhao H, et al. BatGPT: a bidirectional autoregessive talker from generative pre-trained transformer. 2023. ArXiv:2307.00360

75 Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. 2018. ArXiv:1810.04805

76 Liang D, Gonen H, Mao Y, et al. XLM-V: overcoming the vocabulary bottleneck in multilingual masked language models. 2023. ArXiv:2301.10472

77 Chi Z, Dong L, Wei F, et al. InfoXLM: an information-theoretic framework for cross-lingual language model pre-training. 2020. ArXiv:2007.07834

78 Goyal N, Gao C, Chaudhary V, et al. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. Trans Assoc Comput Linguistics, 2022, 10: 522–538

79 Liu X, Zhang C, Chen X, et al. CLTS: a new chinese long text summarization dataset. In: Proceedings of CCF International Conference on Natural Language Processing and Chinese Computing. Springer, 2020. 531–542

80 Coster W, Kauchak D. Simple english wikipedia: a new text simplification task. In: Proceedings of Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, 2011. 665–669

81 Mostafazadeh N, Chambers N, He X, et al. A corpus and cloze evaluation for deeper understanding of commonsense stories. In: Proceedings of Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, 2016. 839–849

82 Lin C Y. ROUGE: a package for automatic evaluation of summaries. In: Proceedings of Text Summarization Branches Out. Barcelona, 2004. 74–81

83 Post M. A call for clarity in reporting BLEU scores. In: Proceedings of Proceedings of the Third Conference on Machine Translation: Research Papers, 2018. 186–191

84 Li J, Galley M, Brockett C, et al. A diversity-promoting objective function for neural conversation models. 2015. ArXiv:1510.03055

85 Shao Z, Huang M, Wen J, et al. Long and diverse text generation with planning-based hierarchical variational model. 2019. ArXiv:1908.06605

86 Pillutla K, Swayamdipta S, Zellers R, et al. MAUVE: measuring the gap between neural text and human text using divergence frontiers. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. Red Hook, 2021. 4816–4828

87 Wang B, Komatsuzaki A. GPT-J-6B: a 6 billion parameter autoregressive language model. https://github.com/kingoflolz/

mesh-transformer-jax, May 2021
88 Alayrac J B, Donahue J, Luc P, et al. Flamingo: a visual language model for few-shot learning. In: Proceedings of 36th Conference on Neural Information Processing Systems. 2022. 23716–23736
89 Zheng L, Chiang W L, Sheng Y, et al. Judging LLM-as-a-judge with mT-bench and chatbot arena. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. New Orleans, 2023. 46595–46623
90 Rawte V, Sheth A, Das A. A survey of hallucination in large foundation models. 2023. ArXiv:2309.05922
91 Zhang J, Qian K, Liu Z, et al. DialogStudio: towards richest and most diverse unified dataset collection for conversational AI. In: Proceedings of Findings of the Association for Computational Linguistics: EACL 2024. St. Julian's, 2024. 2299–2315
92 Ji Y, Deng Y, Gong Y, et al. BELLE: be everyone's large language model engine. https://github.com/LianjiaTech/BELLE, 2023
93 Luo Z, Xu C, Zhao P, et al. WizardCoder: empowering code large language models with Evol-Instruct. 2023. ArXiv:2306.08568
94 Li X, Yu P, Zhou C, et al. Self-alignment with instruction backtranslation. 2023. ArXiv:2308.06259
95 Conover M, Hayes M, Mathur A, et al. Free dolly: introducing the world's first truly open instruction-tuned LLM. Company Blog, 2023
96 Zhou C, Liu P, Xu P, et al. LIMA: less is more for alignment. 2023. ArXiv:2305.11206
97 Schick T, Dwivedi-Yu J, Dessì R, et al. Toolformer: language models can teach themselves to use tools. 2023. ArXiv:2302.04761
98 Wu C, Yin S, Qi W, et al. Visual ChatGPT: talking, drawing and editing with visual foundation models. 2023. ArXiv:2303.04671
99 Zhou W, Zhang S, Gu Y, et al. UniversalNER: targeted distillation from large language models for open named entity recognition. 2023. ArXiv:2308.03279
100 Liu J, Pasupat P, Cyphers S, et al. Asgard: a portable architecture for multilingual dialogue systems. In: Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013. 8386–8390
101 Liu Z, Xu Y, Yu T, et al. CrossNER: evaluating cross-domain named entity recognition. In: Proceedings of Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 13452–13460
102 Wang X, Zhou W, Zu C, et al. InstructUIE: multi-task instruction tuning for unified information extraction. 2023. ArXiv:2304.08085