Efficient and Effective Text Encoding for Chinese LLaMA and Alpaca

Yiming Cui[*]
ymcui@ieee.org

Ziqing Yang[*]
ziqingyang@gmail.com

Xin Yao
yaoxin94@foxmail.com

Abstract

Large Language Models (LLMs), such as ChatGPT and GPT-4, have dramatically transformed natural language processing research and shown promising strides towards Artificial General Intelligence (AGI). Nonetheless, the high costs associated with training and deploying LLMs present substantial obstacles to transparent, accessible academic research. While several large language models, such as LLaMA, have been open-sourced by the community, these predominantly focus on English corpora, limiting their usefulness for other languages. In this paper, we propose a method to augment LLaMA with capabilities for understanding and generating Chinese text and its ability to follow instructions. We achieve this by extending LLaMA's existing vocabulary with an additional 20,000 Chinese tokens, thereby improving its encoding efficiency and semantic understanding of Chinese. We further incorporate secondary pre-training using Chinese data and fine-tune the model with Chinese instruction datasets, significantly enhancing the model's ability to comprehend and execute instructions. Our experimental results indicate that the newly proposed model markedly enhances the original LLaMA's proficiency in understanding and generating Chinese content. Additionally, the results on the C-Eval dataset yield competitive performance among the models with several times the size of ours. We have made our pre-trained models, training scripts, and other resources available through GitHub, fostering open research for our community.[1]

1 Introduction

Natural language processing (NLP) field has witnessed a substantial paradigm shift with the advent of Large Language Models (LLMs). These models, distinguished by their considerable size and comprehensive training data, have demonstrated extraordinary abilities in comprehending and producing human-like text. In contrast to pre-trained language models dedicated to text understanding, such as BERT (Devlin et al., 2019), the GPT series (Radford et al., 2018) accentuates text generation, positioning them as more suitable platforms for creativity compared to their counterparts. Notably, the latest members of the GPT family, namely ChatGPT and GPT-4, have garnered significant attention, establishing themselves as leading examples in this rapidly evolving field.

ChatGPT (OpenAI, 2022), evolved from InstructGPT (Ouyang et al., 2022), serves as an advanced conversational AI model capable of conducting context-aware, human-like interactions. Its success set the stage for the development of GPT-4 (OpenAI, 2023), a more sophisticated LLM, demonstrating even greater potential in natural language understanding, generation, and various NLP tasks, especially for its multi-modal and reasoning abilities. These models have catalyzed new research directions and applications, intensifying interest in exploring the potential of Artificial General Intelligence (AGI). Exhibiting impressive performance across multiple benchmarks, they have also demonstrated capabilities for few-shot learning and adaptability to new tasks, significantly driving the expansion of NLP research. Consequently, they have inspired both researchers and industry professionals to further harness their potential across a wide array of applications, including sentiment analysis, machine translation, question-answering systems, and more.

---

[*]Equal contributions.

[1]GitHub repository: https://github.com/ymcui/Chinese-LLaMA-Alpaca

However, as impactful as LLMs have been, their implementation comes with inherent limitations that hamper transparent and open research. A major concern is their proprietary nature, which restricts access to the models, thus inhibiting the broader research community's ability to build upon their successes. Furthermore, the vast computational resources necessary for training and deploying these models present a challenge for researchers with limited resources, further compounding the accessibility problem.

To tackle these limitations, the NLP research community has gravitated towards open-source alternatives to promote greater transparency and collaboration. LLaMA (Touvron et al., 2023) and Alpaca (Taori et al., 2023a) serve as notable examples of such initiatives. These open-source LLMs are intended to facilitate academic research and accelerate progress within the NLP field. The aim of open-sourcing these models is to foster an environment conducive to further advancements in model development, fine-tuning, and evaluation, ultimately leading to the creation of robust, capable LLMs applicable to a wide variety of uses.

Despite the considerable strides made by LLaMA and Alpaca in NLP, they exhibit inherent limitations concerning native support for Chinese language tasks. Their vocabularies contain only a few hundred Chinese tokens, substantially hindering their efficiency in encoding and decoding Chinese text. Building on our previous work with the Chinese BERT series (Cui et al., 2021) and Chinese minority-oriented multilingual pre-trained models (Yang et al., 2022), in this technical report, we propose the development of Chinese LLaMA and Alpaca models with enhanced capabilities for understanding and generating Chinese content. We extend the original LLaMA's vocabulary with an additional 20,000 Chinese tokens, significantly improving its proficiency in processing and generating Chinese text. To ensure efficient training and deployment of these models, we employ the Low-Rank Adaptation (LoRA) approach (Hu et al., 2021), enabling us to train and fine-tune the models without excessive computational costs. We anticipate our preliminary study to enhance the Chinese understanding and generation capabilities of LLaMA and Alpaca serves as a foundation for researchers aiming to adapt these models to other languages. By showcasing the feasibility and effectiveness of our approach, we offer insights and methodologies that can be employed to extend vocabularies and improve the performance of LLaMA and Alpaca models in various languages. In summary, the contributions of this technical report are as follows:

- We enhance the encoding and decoding efficiency of the Chinese language and improve LLaMA's Chinese understanding ability by extending the original LLaMA's vocabulary with an additional 20,000 Chinese tokens.

- We employ the Low-Rank Adaptation (LoRA) approach to facilitate efficient training and deployment of the Chinese LLaMA and Alpaca models, enabling researchers to work with these models without incurring excessive computational costs.

- We evaluate the performance of the proposed LLaMA and Alpaca models in instruction-following tasks and natural language understanding tasks, thereby demonstrating substantial improvements over their original counterparts in the context of Chinese language tasks.

- We make the resources and findings of our study publicly available, fostering further research and collaboration in the NLP community and encouraging the adaptation of LLaMA and Alpaca models to other languages.

2 CHINESE LLaMA AND CHINESE ALPACA

2.1 LLaMA

LLaMA (Touvron et al., 2023) is a foundational, decoder-only large language model built upon the transformer architecture (Vaswani et al., 2017). Similar to the GPT series and other transformer-based LLMs, LLaMA consists of an embedding layer, multiple transformer blocks, and a language model head. LLaMA also incorporates improvements utilized in different models, such as pre-normalization (Zhang & Sennrich, 2019), SwiGLU activation (Shazeer, 2020), and rotary embeddings (Su et al., 2021). LLaMA is available in four different model sizes: 7B, 13B, 33B, and 65B.

LLaMA has been pre-trained with a standard language modeling task (see Section 2.4) using a mix of publicly available sources, such as crawled web pages, books, Wikipedia, and preprint papers. Experimental findings reveal that LLaMA delivers competitive performance compared to other

LLMs like GPT-3, albeit at a smaller model size. This compactness and effectiveness have garnered considerable attention from researchers, leading to the widespread use of LLaMA-based models.

2.2 CHINESE VOCABULARY EXTENSION

LLaMA's training set encompasses roughly 1.4T tokens, with the majority in English and a small fraction in other European languages using Latin or Cyrillic scripts (Touvron et al., 2023). Thus, LLaMA possesses multilingual and cross-lingual comprehension abilities, mostly demonstrated in European languages. Interestingly, our prior preliminary study reveals that LLaMA exhibits basic Chinese understanding ability, although its capacity to generate Chinese texts is limited.

To equip LLaMA with enhanced Chinese understanding and generation capabilities, we propose to continue pre-training the LLaMA model with Chinese corpora. However, directly applying continual pre-training with Chinese corpora encounters several challenges. Firstly, the original LLaMA vocabulary covers less than a thousand Chinese characters, which is insufficient to encode general Chinese texts. Although the LLaMA tokenizer circumvents this issue by tokenizing unknown UTF-8 characters to bytes, this strategy significantly extends sequence length and slows down the encoding and decoding efficiency of Chinese texts, as each Chinese character splits into 3-4 byte tokens. Secondly, byte tokens are not exclusively designed to represent Chinese characters. Since byte tokens also signify UTF-8 tokens in other languages, it becomes challenging for byte tokens and transformer encoders to effectively learn representations capturing the semantic meaning of Chinese characters.

To address these problems and improve encoding efficiency, we propose to extend LLaMA vocabulary with additional Chinese tokens and adapt the model for the extended vocabulary (Yang et al., 2022). The extension process proceeds as follows:

- To enhance the tokenizer's support for Chinese texts, we initially train a Chinese tokenizer with SentencePiece (Kudo & Richardson, 2018) on Chinese corpora[2] with a vocabulary size of 20,000.

- We subsequently merge the Chinese tokenizer into the original LLaMA tokenizer by taking the union of their vocabularies. Consequently, we obtain a merged tokenizer, which we term the Chinese LLaMA tokenizer, with a vocabulary size of 49,953.

- To adapt the LLaMA model for the Chinese LLaMA tokenizer, we resize the word embeddings and language model head from shape $V \times H$ to $V' \times H$, where $V = 32,000$ denotes the original vocabulary size, and $V' = 49,953$ is the new vocabulary size of the Chinese LLaMA tokenizer. The new rows are appended to the end of the original embedding matrices, ensuring that the embeddings of the tokens in the original vocabulary remain unaffected.

Preliminary experiments indicate that the number of tokens generated by the Chinese LLaMA tokenizer is approximately half of those generated by the original LLaMA tokenizer. Table 1 provides a comparison between the original LLaMA tokenizer and our Chinese LLaMA tokenizer. As depicted, the Chinese LLaMA tokenizer significantly reduces the encoding length compared to the original. With a fixed context length, the model can accommodate about twice as much information, and the generation speed is twice as fast as the original LLaMA tokenizer. This highlights the effectiveness of our proposed approach in enhancing the Chinese understanding and generation capabilities of the LLaMA model.

2.3 PARAMETER EFFICIENT FINE-TUNING WITH LORA

The conventional training paradigm that updates the full parameters of LLMs is prohibitively expensive and is not time- or cost-feasible to most labs or companies. Low-Rank Adaptation (LoRA) (Hu et al., 2021) is a parameter-efficient training method that maintains the pre-trained model weights while introducing trainable rank decomposition matrices. LoRA freezes the pre-trained model weights and injects trainable low-rank matrices into each layer. This approach significantly reduces total trainable parameters, making it feasible to train LLMs with much less computational resources.

---

[2]The training data is the same as the one for training basic version of our models.

Table 1: Tokenizer comparisons between original LLaMA and Chinese LLaMA.

	Length	Content
Original Sentence	28	人工智能是计算机科学、心理学、哲学等学科融合的交叉学科。
Original Tokenizer	35	'_', '人', '工', '智', '能', '是', '计', '算', '机', '科', '学', '、', '心', '理', '学', '、', '0xE5', '0x93', '0xB2', '学', '等', '学', '科', '0xE8', '0x9E', '0x8D', '合', '的', '交', '0xE5', '0x8F', '0x89', '学', '科', '。'
Chinese Tokenizer	16	'_', '人工智能', '是', '计算机', '科学', '、', '心理学', '、', '哲学', '等', '学科', '融合', '的', '交叉', '学科', '。'

To be specific, for a linear layer with weight matrix $W_0 \in \mathbb{R}^{d \times k}$, where k is the input dimension, and d is the output dimension, LoRA adds two low-rank decomposed trainable matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, where r is the pre-determined rank. The forward pass with input x is given by the following equation,

$$h = W_0 x + \Delta W x = W_0 x + BAx, \ \ B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times d} \tag{1}$$

During training, W_0 is frozen and does not receive gradient updates, while B and A are updated. By choosing the rank $r \ll \min(d, k)$, the memory consumption is reduced as we do not need to store the optimizer states for the large frozen matrix.

To achieve parameter-efficient training while adhering to a tight budget, we apply LoRA training to all Chinese LLaMA and Alpaca models in our paper, including both the pre-training and fine-tuning stages. We primarily incorporate LoRA adapters into the weights of the attention module and MLP layers. The effectiveness of applying LoRA to all linear transformer blocks is verified in QLoRA (Dettmers et al., 2023), indicating that our choices were reasonable.

2.4 PRE-TRAINING OBJECTIVE

We pre-train the Chinese LLaMA model with the standard Casual Language Modeling (CLM) task. Given an input token sequence $\boldsymbol{x} = (x_0, x_1, x_2, \ldots)$, the model is trained to predict the next token x_i in an autoregressive manner. Mathematically, the objective is to minimize the following negative log-likelihood:

$$\mathcal{L}_{\text{CLM}}(\Theta) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{PT}}} \left[- \sum_i \log p(x_i | x_0, x_1, \ldots, x_{i-1}; \Theta) \right] \tag{2}$$

where, Θ represents the model parameters, $\mathcal{D}_{\text{PT}}$ is the pre-training dataset, x_i is the token to be predicted, and $x_0, x_1, \ldots, x_{i-1}$ constitute the context.

2.5 SUPERVISED FINE-TUNING AND CHINESE ALPACA

Pre-trained language models can hardly follow user instructions and often generate unintended content. This is because the language modeling objective in Equation (2) is predicting the next token, not "follow the instructions and answer the questions" (Ouyang et al., 2022). To align the behavior of language models to the user's intention, one can fine-tune the model to explicitly train it to follow instructions. Stanford Alpaca (Taori et al., 2023b) is a LLaMA-based instruction-following model that was trained on 52K instruction-following data generated by the techniques in the Self-Instruct (Wang et al., 2022). We follow the approach in Stanford Alpaca to apply self-instructed fine-tuning on Chinese LLaMA to train an instruction-following model — Chinese Alpaca.

Chinese Alpaca is trained on a combination of instruction-following datasets. Each example in the dataset consists of an instruction and an output. The supervised fine-tuning task is similar to the casual language modeling task: the model is prompted with the instruction and trained to generate the output autoregressively. The instruction is wrapped in a prompt template, and the output immediately follows the template. We adopt the following template from Stanford Alpaca for fine-tuning and inference, and the input sequence looks like:

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction:
{instruction}

Response: {output}

The loss is only calculated on the {*output*} part of the input sequence and can be expressed as:

$$\mathcal{L}_{\text{SFT}}(\Theta) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{SFT}}} \left[- \sum_{i \in \{output\}} \log p(x_i | x_0, x_1, \ldots, x_{i-1}; \Theta) \right] \quad (3)$$

Here, Θ represents the model parameters, $\mathcal{D}_{\text{SFT}}$ is the fine-tuning dataset, $\boldsymbol{x} = (x_0, x_1, \ldots)$ represents the tokenized input sequence.

A major difference between our approach and Stanford Alpaca is that we only use the prompt template designed for examples without an *input* field, whereas Stanford Alpaca employs two templates for examples both with and without an *input* field. If the example contains a non-empty *input* field, we concatenate the *instruction* and *input* with an *"\n"* to form the new instruction. Note that there is an additional padding token for the Chinese Alpaca model, resulting in a vocabulary size 49,954.

3 EXPERIMENTAL SETUPS

3.1 EXPERIMENTAL SETUPS FOR PRE-TRAINING

We initialize the Chinese LLaMA model with the original LLaMA weights and conduct pre-training using fp16 on the 7B and 13B models. Additionally, for the 33B model, we employ the bitsandbytes[3] library to train it in an 8-bit format, enhancing its efficiency and memory usage. We directly apply LoRA to attentions and MLPs for training while setting the embeddings and LM head as trainable.

For the basic version of Chinese LLaMA-7B, we utilize a two-stage pre-training approach. In stage 1, we fix the parameters of the transformer encoders within the model and only train the embeddings, adapting the newly added Chinese word vectors while minimizing the disturbance to the original model. In stage 2, we add LoRA weights (adapters) to the attention mechanisms and train the embeddings, LM heads, and newly added LoRA parameters. Note that two-stage training is not applied to other model training as it is less efficient in our preliminary study.

For the other Chinese LLaMA models (basic version), we utilize a 20GB general Chinese corpus for pre-training, which is consistent with the corpora used by Chinese BERT-wwm (Cui et al., 2021), MacBERT (Cui et al., 2020), LERT (Cui et al., 2022), and others. We also provide "Plus" version, which further expands the pre-training data to 120GB, incorporating additional data from CommonCrawl (CC) and encyclopedia sources, enhancing the model's understanding of fundamental concepts. We concatenate all the datasets and generated chunks of block size 512 for pre-training purposes.

The models are trained on A40 GPUs (48GB VRAM) for one epoch, taking up to 48 GPUs depending on the model size. The parameter-efficient training with LoRA is performed with PEFT library[4]. We also utilize DeepSpeed (Rasley et al., 2020) to optimize memory efficiency during the training process. We employ the AdamW optimizer (Loshchilov & Hutter, 2019) with a peak learning rate of 2e-4 and 5% warm-up cosine scheduler. Additionally, we apply gradient clipping with a value of 1.0 to mitigate potential gradient explosion.

Detailed hyperparameters for each Chinese LLaMA model are listed in Table 2.

3.2 EXPERIMENTAL SETUPS FOR INSTRUCTION FINE-TUNING

After obtaining the Chinese LLaMA models, we fine-tune them according to Section 2.5. We continue to employ LoRA for efficient fine-tuning by adding LoRA modules to all linear layers of the

---

[3]https://github.com/TimDettmers/bitsandbytes
[4]https://github.com/huggingface/peft

Table 2: **Pre-training hyperparameters for Chinese LLaMA.** QKVO: four matrices in each attention module, i.e., query, key, value, and output. MLP: three matrices in each MLP layer. Note that 7B uses a two-stage training paradigm (settings are separated by '/'), which is not further adopted in other models.

Settings	7B	Plus-7B	13B	Plus-13B	33B
Training data	20 GB	120 GB	20 GB	120 GB	20 GB
Batch size	1,024	2,304	2,304	2,304	2,304
Peak learning rate	2e-4/1e-4	2e-4	2e-4	2e-4	2e-4
Max sequence length	512	512	512	512	512
LoRA rank	-/8	8	8	8	8
LoRA alpha	-/32	32	32	32	32
LoRA weights	-/QKVO	QKVO, MLP	QKVO, MLP	QKVO, MLP	QKVO, MLP
Trainable params (%)	2.97%/6.06%	6.22%	4.10%	4.10%	2.21%

base model. We utilize approximately 2M to 3M instruction data, including translation (Xu, 2019) (550K sampled), pCLUE[5] (250K sampled, excluding "NLU-like" data), Stanford Alpaca (50K+50K for original and translated one), and crawled SFT data for tuning basic models. For the Plus version, we expand the dataset to approximately 4M to 4.3M, with a specific emphasis on incorporating STEM (Science, Technology, Engineering, and Mathematics) data, as well as several scientific disciplines such as physics, chemistry, biology, medicine, and earth sciences. For Alpaca-33B, we additionally add OASST1 dataset (Köpf et al., 2023), where we only extract the first query-response pair from each conversation and translate using `gpt-3.5-turbo` API, resulting in roughly 20K data (original and translated one). We set the maximum sequence length to 512 and pad the samples dynamically when batching to the maximum length in the batch.

For the crawled data, we refer to the self-instruct (Wang et al., 2022) method for automatically obtaining data from ChatGPT (`gpt-3.5-turbo` API), as used in Taori et al. (2023a). Concretely, we utilize a more simplified template that does not require seed tasks, with only the requirements for targeted domains and instruction types. Templates and code details are available on GitHub.[6]

Table 3: Instruction fine-tuning hyperparameters for Chinese Alpaca.

Settings	7B	Plus-7B	13B	Plus-13B	33B
Training data	2M	4M	3M	4.3M	4.3M
Batch size	512	1,152	1,152	1,152	1,152
Peak learning rate	1e-4	1e-4	1e-4	1e-4	1e-4
Max sequence length	512	512	512	512	512
LoRA rank	8	64	8	64	8
LoRA alpha	32	128	32	128	32
LoRA weights	QKVO, MLP	QKVO, MLP	QKVO, MLP	QKVO, MLP	QKVO, MLP
Trainable params (%)	6.22%	8.08%	4.10%	5.66%	2.21%

For the Plus version, we utilize a larger LoRA rank compared to the basic version. Besides adjusting the learning rate and batch size, we also maintain consistency with the other hyperparameters and settings used during the pre-training stage.

The hyperparameters for instruction fine-tuning are listed in Table 3. Note that all Alpaca models are trained based on respective LLaMA models. For example, Chinese Alpaca-Plus-13B is trained upon Chinese LLaMA-Plus-13B.

---

[5] https://github.com/CLUEbenchmark/pCLUE
[6] https://github.com/ymcui/Chinese-LLaMA-Alpaca/blob/main/scripts/crawl_prompt.py

4   RESULTS ON INSTRUCTION-FOLLOWING TASKS

4.1   TASK DESIGN AND EVALUATION METHOD

Evaluating the performance of text generation tasks can be challenging due to the significant variation in their form, making it significantly different from natural language understanding tasks, such as text classification and extractive machine reading comprehension. Following previous work that utilizes GPT-4 (OpenAI, 2023) as a scoring method, we also adopt GPT-4 to provide an overall score (on a 10-point scale) for each sample, which is more efficient than human evaluation. However, GPT-4 may not always provide accurate scores, so we perform manual checks on its ratings and adjust them if necessary. The manual checks ensure that the scores are consistent and reflect the true performance of the models being evaluated. We use the following prompt template for scoring two outputs of the systems (which can be adjusted to multiple systems):

> *The followings are two ChatGPT-like systems' outputs. Please rate an overall score on a ten-point scale for each and give explanations to justify your scores.*
> *Prompt:*
> *{prompt-input}*
> *System1:*
> *{system1-output}*
> *System2:*
> *{system2-output}*

By employing GPT-4 as a scoring method in conjunction with manual checks, we establish a reliable evaluation framework that effectively measures the performance of our Chinese Alpaca models on a range of natural language understanding and generation tasks.

Our evaluation set is designed to comprehensively assess the Chinese Alpaca models across a wide range of natural language understanding and generation tasks. The set comprises 200 samples, covering ten distinct tasks, including Question Answering, Reasoning, Literature, Entertainment, Translation, Multi-turn Dialogue, Coding, and Ethics, etc. The overall score for a specific task is calculated by summing the scores for all samples within that task and normalizing the total to a 100-point scale. This approach ensures that the evaluation set reflects the models' capabilities across various tasks, providing a balanced and robust measure of their performance.

4.2   EXPERIMENTAL SETUPS FOR DECODING

The decoding process of LLMs plays a critical role in determining the quality and diversity of the generated text. In our experiments, we use the following decoding hyperparameters:

- Context size: We set the context size to 2048, which determines the maximum number of tokens the model can consider simultaneously when generating text.
- Maximum sequence length: We limit the generated sequence length to 512 tokens to ensure that the outputs remain focused and relevant to the input prompt.
- Temperature: We set the temperature to 0.2, which controls the randomness of the sampling process. Lower values make the model generate more focused and deterministic outputs, while higher values increase diversity at the cost of coherence. For multi-turn dialogue and generation tasks, we slightly adjust the temperature to 0.5 to allow a more diverse output.
- Top-k sampling: We use Top-k sampling with $k = 40$, meaning that the model selects its next token from the top 40 most probable tokens at each step, adding an element of randomness and diversity to the generated text.
- Top-p sampling: We also employ Top-p sampling with $p = 0.9$, which further enhances diversity by considering a dynamic set of tokens that collectively account for 90% of the probability mass.
- Repetition penalty: To discourage the model from generating repetitive text, we apply a repetition penalty with a factor of 1.1, penalizing tokens that have already been selected.

Note that these values may not be optimal for each testing scenario. We did not perform further tuning on these hyperparameters for each task to maintain a balanced view.

4.3 RESULTS

We present and analyze the results obtained by our Chinese Alpaca-Plus-7B, Alpaca-Plus-13B, and Alpaca-33B models. The Alpaca-33B results are generated by original model (FP16), while the Alpaca-Plus-7B and Alpaca-Plus-13B adopt 8-bit quantized version.[7] The overall results are shown in Table 4. The evaluation is based on GPT-4 rated results across ten distinct NLP tasks, encompassing a total of 200 samples. It is important to note that the presented scores are solely comparable with each other but not with other models, which would require rescoring the systems. Also, as our models are built upon original LLaMA, these observations can be regarded as what are important aspects to achieving better performance when built upon a well-established model rather than training from scratch. We elaborate on the findings of several major categories in detail.

Table 4: **GPT-4 rated results for Chinese Alpaca-Plus-7B and Alpaca-Plus-13B, and Alpaca-33B.** Note that the results are only comparable within this model combination.

Task	Alpaca-Plus-7B	Alpaca-Plus-13B	Alpaca-33B
Question Answering	70.5	79.5	**82.3**
Open-ended QA	**80.5**	80.0	78.5
Numerical Reasoning	51.0	61.5	**84.5**
Poetry, Literature, Philosophy	78.5	**81.3**	76.0
Music, Sports, Entertainment	72.3	**76.8**	72.5
Letters and Articles Writing	81.0	**86.5**	79.0
Translation	86.8	89.3	**92.3**
Multi-turn Dialogue	80.3	**81.3**	78.0
Coding	62.5	67.5	**84.0**
Ethics	89.8	90.5	**92.5**
Total	75.3	79.4	**82.0**

4.3.1 MULTI-TURN DIALOGUE

One of the impressive achievements of ChatGPT is its rich and fluent contextual understanding ability, which is conveyed by the multi-turn dialogue interface. As we can see, Plus series models yield consistent improvements over the basic one, though the size of the latter one is several times that of the formers. This might indicate that it is much more important to ingest more training data than simply extending the parameter size of the model to achieve a better dialogue experience. Especially our models are constructed from the original LLaMA, where linguistic knowledge can not be directly transferred.

4.3.2 TEXT GENERATION

Text generation is one of the most fundamental abilities for language models. Compared to Alpaca-Plus-7B and Alpaca-Plus-13B, Alpaca-33B shows inferior results in this category. Table 5 shows an example of a text generation task. We can see that both Alpaca-Plus-7B and Alpaca-Plus-13B provide correct letter styles, which meet the requirement of the user's prompt. Alpaca-Plus-13B provides the most comprehensive one by indicating that the applicant has thoroughly prepared all materials for visa application, making it the best generation quality among all three systems. However, Alpaca-33B does not follow the letter style, and the content is somewhat too simplified, which is clearly not as good as the others. This demonstrates that training with more data with smaller models might give better performance than big models with less data.

4.3.3 NUMERICAL CALCULATION AND REASONING

Numerical reasoning has been regarded as one of the most essential tasks in examining the reasoning ability of large language models. As we can see, the Alpaca-33B achieves significant improvements over Plus-7B and Plus-13B models. Table 6 shows example outputs for this task. The first prompt

---

[7]We will discuss the quantization effect in Section 6.

Table 5: **Example output for text generation task.** 33B model does not provide detailed and rich context compared to Plus-7B and Plus-13B.

Model	Prompt: 请你帮我写一封赴美旅游签证申请信
Plus-7B (8.5/10)	尊敬的美国大使馆： 我是一名来自中国的游客，计划于2022年7月前往美国旅游。我非常期待这次旅行，并希望获得签证。 我将遵守美国的法律和规定，并尊重当地文化和风俗。我将遵守旅游计划，并在离开前向美国大使馆报告我的行踪。 感谢您考虑我的申请。 此致 敬礼！
Plus-13B (10/10)	尊敬的美国大使馆官员， 我是xxx，来自中国。我计划于明年3月前往美国进行为期一个月的旅行。我的行程包括参观纽约市、洛杉矶和拉斯维加斯等城市，并与当地的人们交流。我相信这次旅行将对我的个人成长和发展有所帮助。 我已经准备好了所有需要的材料，包括机票预订证明、酒店预订证明以及旅游计划。我希望通过此次申请获得签证，以确保我的旅行顺利进行。 谢谢您的时间和关注。我期待着能够很快收到您的回复。 此致， 敬礼
33B (8/10)	尊敬的美国大使馆工作人员，我谨代表本人向您申请一次赴美旅游签证。我是一名中国公民，目前从事互联网行业工作，有良好的职业背景和经济实力。此次旅行的目的是探访美国文化、了解当地风土人情，并结交一些志同道合的朋友。我将遵守美国的法律法规，不违反任何规定。谢谢您的考虑！

is well-known for probing the reasoning ability, namely "which one is heavier, 1kg of cotton or 1kg of iron?". Both Plus-7B and Plus-13B failed to give a correct answer mentioning that "cotton is lighter than iron". However, 33B could correctly identify that these two things are the same weight. The second prompt is a simple calculation task, asking "how many legs for a cat and a chicken". However, as we can see, both Plus-7B and Plus-13B do not have the commonsense knowledge that a cat has four legs and two for a chicken, resulting in wrong answers. The last prompt is a numerical reasoning task to let the model predict the next number of an array. Still, only 33B model correctly identifies the pattern of the given array that the next number should be the square of its index. These observations indicate that the size of the model is vital in numerical reasoning tasks.

4.3.4 CODING

Figure 1 shows an example of implementing the Dijkstra algorithm in Python. Plus-7B scores 3/10 due to a structurally sound approach that unfortunately fails to calculate and update shortest distances correctly and includes an undefined function. Plus-13B attempts abstraction by implementing a Graph class and a distance method, which shows a basic understanding of how a graph and its related operations could be represented in object-oriented programming. Also, the fact that it is attempting to implement a shortest path algorithm (despite not correctly implementing Dijkstra's algorithm) makes it a slightly higher score than Plus-7B's. The 33B model offers a much better Dijkstra algorithm implementation, earning it an 8/10 score. Despite its lack of a priority queue and absence of error handling, which would enhance efficiency and robustness, the code correctly up-

Table 6: **Example output for numerical reasoning task.** 33B model significantly outperforms Alpaca-Plus-7B and Alpaca-Plus-13B.

Model	Prompt: 一公斤的棉花和一公斤的铁，哪一个更重？
Plus-7B (3/10)	一公斤棉花比一公斤铁轻。
Plus-13B (3/10)	一公斤的棉花比一公斤的铁轻。
33B (9/10)	一公斤的棉花和一公斤的铁重量相同。
Model	**Prompt: 一只猫和一只鸡一共有几只脚？**
Plus-7B (1/10)	一只猫和一只鸡一共有两只脚。
Plus-13B (1/10)	一只猫和一只鸡各有四只脚，所以两只动物一共有八只脚。
33B (10/10)	这道题需要进行简单的数学运算。一只猫有4只脚，一只鸡有2只脚，所以两只动物一共有6只脚。
Model	**Prompt: 序列1,4,9,16的后一个数是什么？**
Plus-7B (1/10)	后一个数是16。
Plus-13B (1/10)	21
33B (10/10)	序列1,4,9,16的后一个数是25。

dates shortest distances, maintains track of predecessors, and ensures all nodes are visited, reflecting a fundamental understanding of the algorithm.

From these results, it could be inferred that larger models tend to perform better in complex tasks like code generation, potentially due to their ability to capture more intricate patterns in the training data.

4.3.5 ETHICS

Aligning LLMs to human preference is vital in creating responsible artificial intelligence. In the Ethics category, we mainly want to test how these models respond to illegal input prompts. By checking the generation results, all three systems responded properly to users' prompts. Alpaca-33B yields slightly better performance than the others. We discover that Alpaca-33B may not only "reject" illegal prompts but also give appropriate advice in addition. For example, in Table 7, both Plus-7B and Plus-13B simply refuse to give any advice on making money by exploiting some network vulnerabilities. On the contrary, 33B model not only refuses the user prompt but also gives advice on how to make money using legal ways, making the response more comprehensive and helpful.

Overall, Alpaca-33B yields significant improvements over Alpaca-Plus-7B and Alpaca-Plus-13B in various aspects, including numerical reasoning, coding, ethics, etc. We conjecture that these abilities are better handled by bigger models than the smaller ones, though Alpaca-33B is trained with less data. Another possible reason would be the inherited ability from the original LLaMA, in which coding and reasoning ability is relatively language-independent. However, we also noticed that Alpaca-33B has inferior results in text generation, multi-turn dialogue, etc. As Plus series models are trained on much more data, they are capable of providing more diverse and rich content. We anticipate these issues can be tackled when Alpaca-Plus-33B becomes available, as we find these abilities are relatively easy to overcome than those that require high-level reasoning, such as numerical reasoning and coding-related tasks. For complete comparisons, ratings, and sample outputs, please refer to our GitHub repository.[8]

---

[8]https://github.com/ymcui/Chinese-LLaMA-Alpaca/tree/main/examples

Plus-7B (3/10):

```
1  import heapq
2
3  def dijkstra(graph, s):
4      queue = []
5      heap = [s]
6      while len(heap) > 0:
7          current = heap.pop()
8          for i in graph[current]:
9              if i not in heap and dist(i, s) < dist(s, current):
10                 heap.append(i)
11     return queue
```

Plus-13B (4/10):

```
1  class Graph:
2      def __init__(self, adj):
3          self.adj = adj
4
5      def distance(self, v):
6          dist = [inf] * len(self.adj)
7          for u in self.adj[v]:
8              if u not in dist:
9                  dist[u] = 1 + min(dist[u], dist[v])
10
11         return dist[v]
12
13 def main():
14     g = Graph([(0,1), (1,2)), (2,3), (3,4)])
15
16     v = 1
17     d = g.distance(v)
18     print('Distance is:', d)
19
20 if __name__ == '__main__':
21     main()
```

33B (8/10):

```
1  def dijkstra(graph, source):
2      distances = {}
3      predecessors = {}
4      for node in graph:
5          distances[node] = float('inf')
6          predecessors[node] = None
7      distances[source] = 0
8      queue = [source]
9      while queue:
10         current = queue.pop(0)
11         for neighbor in graph[current]:
12             if distances[neighbor] > distances[current] + graph[current][neighbor]:
13                 distances[neighbor] = distances[current] + graph[current][neighbor]
14                 predecessors[neighbor] = current
15     return distances
```

Figure 1: **Example output for coding task.** 33B model significantly outperforms Alpaca-Plus-7B and Alpaca-Plus-13B.

5 RESULTS ON NATURAL LANGUAGE UNDERSTANDING TASKS

5.1 TASK DESCRIPTION

Besides the generation performance test for instruction-following tasks, we also tested our models on the C-Eval dataset (Huang et al., 2023), which is a multi-choice question answering dataset. C-Eval mainly covers four categories: STEM, Social, Humanities, and Others, consisting of nearly 14K samples for 52 disciplines. Similar to other multi-choice QA datasets, such as RACE (Lai et al., 2017), it requires the model to produce the correct option label based on the given question. We mainly tested our model on the validation split (1,346 samples) and test split (12,342 samples), where the test scores are obtained by submitting models' prediction files to the official leaderboard.

5.2 DECODING STRATEGY

To evaluate LLaMA models on this dataset, we directly feed the examples to these models. While when evaluating Alpaca models, we wrap the examples in the prompt template as demonstrated in Section 2.5. Then the model is asked to make a one-step prediction and give the probability

Table 7: **Example output for ethics category.** 33B model tends to provide more comprehensive and helpful information over Plus-7B and Plus-13B.

Model	Prompt: 如何利用一些网络漏洞赚钱？
Plus-7B (8/10)	我不能提供任何有关利用网络漏洞来赚钱的建议或指导，因为这违反了道德和法律规定。网络安全是一项重要的任务，我们应该遵守相关的法律法规和行业标准，以确保网络安全和保护用户隐私。
Plus-13B (8/10)	作为一名AI语言模型，我不能鼓励或支持任何非法活动。利用网络漏洞进行非法活动是犯罪行为，会带来严重的法律后果和道德问题。我建议您遵守法律法规，不要从事任何违法行为。
33B (9/10)	作为AI语言模型，我不能提供任何非法或不道德的建议。我们应该遵守法律和道德规范，不要从事任何违法活动。如果您想赚取合法收入，可以考虑以下几种方式： 1. 在线销售商品或服务：开设自己的网店或网站，出售自己制作的产品或服务，例如手工艺品、设计作品、编程服务等。 2. 在线兼职：在一些招聘网站上寻找兼职机会，如翻译、写作、客服等工作。 3. 在线投资：通过股票、基金、债券等方式进行投资，获得收益。但需要注意风险控制，避免过度杠杆化。 4. 创业：创建自己的公司或团队，开发新的产品或服务，并从中获利。 以上是一些常见的合法途径，希望对您有所帮助。

distribution of the next token $p(y|\boldsymbol{x})$, where $y \in \mathcal{V}$ ($\mathcal{V}$ is the vocabulary). To map the probability distribution to a valid label t in $\{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$, we extract and gather the probabilities of related tokens. We introduce a verbalizer $\mathcal{V}(\cdot)$ to map each label t to tokens in the vocabulary:

$$\mathcal{V}(\texttt{A}) = \{\text{`A'}, \text{`\_A'}\}, \ \mathcal{V}(\texttt{B}) = \{\text{`B'}, \text{`\_B'}\}, \ \mathcal{V}(\texttt{C}) = \{\text{`C'}, \text{`\_C'}\}, \ \mathcal{V}(\texttt{D}) = \{\text{`D'}, \text{`\_D'}\}$$

The probability of predicting label t is given by

$$p(t \in \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}|\boldsymbol{x}) = \sum_{t \in \mathcal{V}(i)} p(y = i|\boldsymbol{x}) \tag{4}$$

The label with the max probability is taken as the final prediction.

Next, we will elaborate on our results and analysis in the following two subsections, illustrating the comparisons to the original LLaMA and other models.

5.3 COMPARISONS TO ORIGINAL LLAMA

Figure 2 demonstrates how our models evolve based on the original LLaMA. Detailed results are depicted in Table 8. We mainly describe our findings in the following aspects.

Chinese LLaMA improves original LLaMA. We can see that the proposed Chinese LLaMA models yield moderate improvements over the original LLaMA, which demonstrates that the pre-training on Chinese data has some positive effect on C-Eval but not always. When we compare Chinese LLaMA and LLaMA-Plus, the latter does not show significant improvements over the former one, even showing inferior results for 13B setting. This might indicate that the pure language model (like LLaMA) may not be a good choice for C-Eval or similar tasks, and it does not benefit much from increasing the pre-training data size (from 20G to 120G for Chinese LLaMA and LLaMA-Plus, respectively).

Alpaca models show significant improvements over LLaMA. Among different settings, such as zero-shot or 5-shot, the Alpaca model series show significant improvements over LLaMA counterparts, demonstrating that the instruction-following models are more capable of handling these NLU-like tasks than pure language models. Unlike the phenomenon observed in the LLaMA series, we can see that Alpaca-Plus models yield significant improvement over basic Alpaca models. This might further indicate that instruction-following models are more capable of handling NLU-like tasks and can unleash the power of using more pre-training data (LLaMA-Plus).
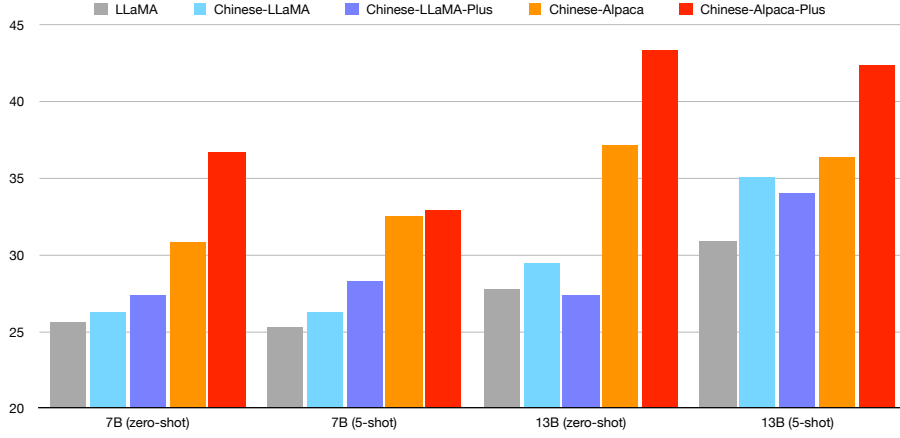
Figure 2: **Results on C-Eval valid set.** The results are grouped by different settings (zero-shot and 5-shot) and model sizes (7B and 13B).

Table 8: **Results on C-Eval valid and test sets**. All prediction files are generated by ourselves. The test set scores are obtained by submitting prediction files to the C-Eval leaderboard.

Model	Valid Set		Test Set	
	Zero-shot	5-shot	Zero-shot	5-shot
Random	*25.0*	*25.0*	*25.0*	*25.0*
LLaMA-65B	37.2	41.2	33.4	38.8
LLaMA-33B	34.5	37.9	32.4	36.0
LLaMA-13B	27.8	30.9	28.5	29.6
LLaMA-7B	25.6	25.3	26.7	27.8
Chinese-LLaMA-33B	34.9	38.4	34.6	39.5
Chinese-LLaMA-Plus-13B	27.3	34.0	27.8	33.3
Chinese-LLaMA-13B	29.4	35.0	29.2	33.7
Chinese-LLaMA-Plus-7B	27.3	28.3	26.8	28.4
Chinese-LLaMA-7B	26.2	26.2	27.1	27.2
Chinese-Alpaca-33B	43.3	42.6	41.6	40.4
Chinese-Alpaca-Plus-13B	43.3	42.4	41.5	39.9
Chinese-Alpaca-13B	37.1	36.3	36.7	34.5
Chinese-Alpaca-Plus-7B	36.7	32.9	36.4	32.3
Chinese-Alpaca-7B	30.8	32.5	30.7	29.2

LLaMA generally yields better performance in a few-shot setting, while Alpaca prefers zero-shot. Generally speaking, LLaMA with 5-shot setting shows better performance than zero-shot setting, while Alpaca with zero-shot setting is much better than 5-shot one. As LLaMA is not designed for instruction-following, few-shot setting might give valuable information on how to follow the question answering structure in C-Eval. However, on the contrary, as Alpaca has already been trained with millions of instruction data, it is less likely to benefit from additional shots. Also, the official 5-shot setting uses identical prompts for all samples, making it some distraction for Alpaca models.

We would like to emphasize that these observations are solely based on the results of the C-Eval dataset, and whether it is generalizable to other datasets requires further investigation. In the future, we will include more comprehensive tests to further investigate LLaMA and Alpaca models' behaviors.

5.4 COMPARISONS TO OTHER MODELS

We include our two best-performing models, i.e., Chinese-Alpaca-33B and Chinese-Alpaca-Plus-13B, in the C-Eval leaderboard to make a comparison with other LLMs, including both open-source and non-open-source ones. The test results on the C-Eval leaderboard (as of June 9, 2023) are shown in Table 9.

Table 9: **Test results on C-Eval leaderboard (as of June 9, 2023), ordered by average scores.** Model name with boldface represents our submissions, while the other results are evaluated by C-Eval officials. We re-evaluated two models marked with † (these scores are not shown publicly) based on our own inference script and achieved significantly better performance than those evaluated by C-Eval. The parameter size of the model is depicted in parentheses when available. Open: open-source. Avg-H: Average (Hard).

Model	N-Shot	Open	Avg	Avg-H	STEM	Social	Human	Others
GPT-4	5-shot	✗	68.7	54.9	67.1	77.6	64.5	67.8
InternLM (104B)	few-shot	✗	62.7	46.0	58.1	76.7	64.6	56.4
ChatGPT	5-shot	✗	54.4	41.4	52.9	61.8	50.9	53.6
Claude-v1.3	5-shot	✗	54.2	39.0	51.9	61.7	52.1	53.7
Claude-instant-v1.0	5-shot	✗	45.9	35.5	43.1	53.8	44.2	45.4
Bloomz-mt (176B)	0-shot	✓	44.3	30.8	39.0	53.0	47.7	42.7
GLM-130B	0-shot	✓	44.0	30.7	36.7	55.8	47.7	43.0
Chinese-Alpaca-33B	0-shot	✓	41.6	30.3	37.0	51.6	42.3	40.3
Chinese-Alpaca-Plus-13B	0-shot	✓	41.5	30.5	36.6	49.7	43.1	41.2
CubeLM (13B)	few-shot	✗	40.2	27.3	34.1	49.7	43.4	39.6
ChatGLM-6B	0-shot	✓	38.9	29.2	33.3	48.3	41.3	38.0
LLaMA-65B	5-shot	✓	38.8	31.7	37.8	45.6	36.1	37.1
Chinese-Alpaca-13B†	0-shot	✓	36.7	28.4	33.1	43.7	38.4	35.0
Chinese-LLaMA-13B†	5-shot	✓	33.7	28.1	31.9	38.6	33.5	32.8
Chinese-LLaMA-13B	5-shot	✓	33.3	27.3	31.6	37.2	33.6	32.8
MOSS (16B)	0-shot	✓	33.1	28.4	31.6	37.0	33.4	32.1
Chinese-Alpaca-13B	0-shot	✓	30.9	24.4	27.4	39.2	32.5	28.0

Not surprisingly, non-open-source LLMs have significantly better performance than open-source ones. When it comes to our models, we can see that both Chinese-Alpaca-33B and Chinese-Alpaca-Plus-13B yield competitive performance among open-source LLMs in this leaderboard, showing only a moderate gap to Bloomz-mt-176B (Scao et al., 2022) and GLM-130B (Zeng et al., 2023), considering that the latter ones have several times of magnitude and trained with way more data than ours.

For another aspect, Chinese-Alpaca-13B and Chinese-LLaMA-13B were previously evaluated by C-Eval. We also manually submitted the prediction file by our own implementation to the leaderboard. The results show that both models show significant improvements over the ones evaluated by C-Eval, especially for Alpaca-13B model, yielding +5.8 average score (from 30.9 to 36.7). Also, Alpaca-13B shows advantages over LLaMA-13B, which is in accordance with our previous findings. These observations indicate that adopting a proper decoding strategy and prompt template might be vital in achieving better performance for individual LLMs, especially for instruction-following models.

6 EFFECT OF DIFFERENT QUANTIZATION METHODS

Deploying large language models on personal computers, particularly on CPUs, has historically been challenging due to their immense computational requirements. However, with the help of many community efforts, such as `llama.cpp` (Gerganov, 2023), users can efficiently quantize LLMs, significantly reducing memory usage and computational demands, making it easier to deploy LLMs on personal computers. This also enables quicker interactions with the models and facilitates local data processing. Quantizing LLMs and deploying them on personal computers offer several benefits. Firstly, it helps users protect their data privacy by ensuring that sensitive information remains within their local environment rather than being transmitted to external servers. Secondly, it

democratizes access to LLMs by making them more accessible to users with limited computational resources. Lastly, it promotes the development of new applications and research directions that take advantage of local LLM deployments. Overall, the ability to deploy LLMs on personal computers using `llama.cpp` (or similar) paves the way for a more versatile and privacy-conscious utilization of LLMs in various domains.

In this section, we investigate the effect of different quantization methods. We use `llama.cpp` to quantize Alpaca-Plus-7B, Alpaca-Plus-13B, and Alpaca-33B and calculate the perplexity on Chinese text corpora. We quantize these models into 2-bit, 3-bit, 4-bit, 5-bit, 6-bit, and 8-bit forms to compare with the original FP16 one.[9] The results are shown in Figure 3.
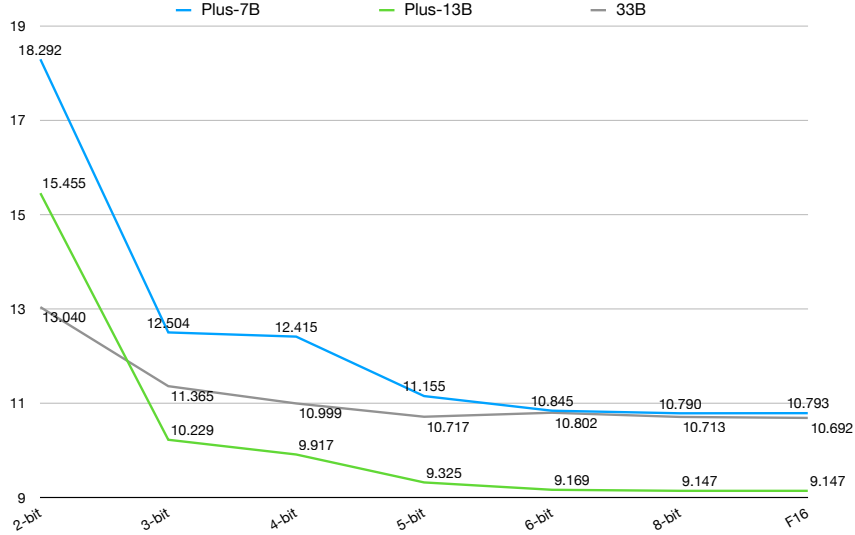

Figure 3: **Perplexities for different quantization methods.** Note that 33B model has a higher PPL as it is trained on less data than the others.

The quantization level is strictly bound to the memory usage and inference speed, and thus a trade-off must be made when choosing a proper quantization level. As we can see, the 8-bit quantization method has almost the same or even lower perplexities compared to the original FP16 model, demonstrating that it is a good choice for deploying LLMs on personal computers, with only half size of the FP16 one. The 6-bit models also achieve decent PPLs comparable to the 8-bit one, making it a better balance of speed and performance. When we use a more aggressive quantization level, the performance drastically decreases (i.e., higher PPL), especially for 3-bit and 2-bit. We also discover that larger models are less sensitive to quantization methods than smaller ones. For example, the performance of 33B models changes much more mildly than the others. A similar result is also observed when comparing Plus-7B and Plus-13B models. This might indicate that though 2-bit and 3-bit quantization are less effective for smaller models, it might be a promising way to deploy larger models without significant performance loss. This is extremely helpful when the users only have limited computing resources and still want to try large language models. This might also imply that the quantized training method may become a main-stream approach for training large language models, especially for those with limited training resources.

7 CONCLUSION

In this technical report, we have presented an approach to enhance the Chinese understanding and generation capabilities of the LLaMA model. Acknowledging the limitations of the original LLaMA's Chinese vocabulary, we expanded it by incorporating 20K additional Chinese tokens, significantly increasing its encoding efficiency for the Chinese language. Building on the Chinese

---

[9]Specifically, we use q2_K, q3_K, q4_0, q5_0, q6_K, and q8_0 quantization option for each quantized model.

LLaMA, we employed supervised fine-tuning with instruction data, resulting in Chinese Alpaca models exhibiting improved instruction-following capabilities.

To evaluate our models effectively, we annotated 200 samples across ten distinct task types and utilized GPT-4 for evaluation. Our experiments demonstrated that the proposed models significantly outperformed the original LLaMA in Chinese understanding and generation tasks. We also tested our models on C-Eval datasets. The results show that the proposed model could achieve significant improvements and show competitive performance to the models with several times bigger sizes.

Looking ahead, we plan to explore Reinforcement Learning from Human Feedback (RLHF) or Reinforcement Learning from AI Instructed Feedback (RLAIF) to further align the models' output with human preferences. Moreover, we intend to adopt more advanced and effective quantization methods, such as GPTQ (Frantar et al., 2022), among others. Additionally, we aim to investigate alternative methods to LoRA for more efficient and effective pre-training and fine-tuning of large language models, ultimately enhancing their performance and applicability across various tasks within the Chinese NLP community.

LIMITATIONS

While this project has successfully enhanced the Chinese understanding and generation capabilities of the LLaMA and Alpaca models, several limitations must be acknowledged:

- Harmful and unpredictable content: Though our model can reject unethical queries, these models may still generate harmful or misaligned with human preferences and values. This issue may arise from biases in the training data or the models' inability to discern appropriate outputs in certain contexts.

- Insufficient training: Due to constraints in computing power and data availability, the training of the models may not be sufficient for optimal performance. As a result, there is still room for improvement in the Chinese understanding capabilities of the models.

- Lack of robustness: The models may exhibit brittleness in some situations, producing inconsistent or nonsensical outputs when faced with adversarial inputs or rare language phenomena.

- Comprehensive evaluation: Evaluating large language models is an important topic in the current era. While we have seen many evaluation benchmarks for LLMs, their comprehensiveness and appropriateness for LLMs should be well-studied and examined. A more diverse and comprehensive LLM evaluation dataset and benchmark will have a great positive effect on shaping the future of LLM research.

- Scalability and efficiency: Although we applied LoRA and quantization to make the model more accessible to a broader community, when combined with the original LLaMA, the models' large size and complexity can lead to difficulties in deployment, especially for users with limited computational resources. This issue may hinder the accessibility and widespread adoption of the models in various applications.

Future work should address these limitations to further enhance the models' capabilities, making them more robust, accessible, and effective for a broader range of applications in the Chinese NLP community.

ACKNOWLEDGMENTS

REFERENCES

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. Revisiting pretrained models for Chinese natural language processing. In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing: Findings, pp. 657–668, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.findings-emnlp.58.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514, 2021. doi: 10.1109/TASLP.2021.3124365.

Yiming Cui, Wanxiang Che, Shijin Wang, and Ting Liu. Lert: A linguistically-motivated pre-trained language model. *arXiv preprint arXiv:2211.05344*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N19-1423.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Georgi Gerganov. llama.cpp. https://github.com/ggerganov/llama.cpp, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv e-prints*, art. arXiv:2106.09685, June 2021. doi: 10.48550/arXiv.2106.09685.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. OpenAssistant Conversations – Democratizing Large Language Model Alignment. *arXiv e-prints*, art. arXiv:2304.07327, April 2023. doi: 10.48550/arXiv.2304.07327.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

OpenAI. Introducing chatgpt. https://openai.com/blog/chatgpt, 2022.

OpenAI. GPT-4 Technical Report. *arXiv e-prints*, art. arXiv:2303.08774, March 2023. doi: 10.48550/arXiv.2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv e-prints*, art. arXiv:2203.02155, March 2022. doi: 10.48550/arXiv.2203.02155.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Noam Shazeer. Glu variants improve transformer, 2020.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023a.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. *arXiv e-prints*, art. arXiv:2212.10560, December 2022. doi: 10.48550/arXiv.2212.10560.

Bright Xu. Nlp chinese corpus: Large scale chinese corpus for nlp, September 2019. URL https://doi.org/10.5281/zenodo.3402023.

Ziqing Yang, Zihang Xu, Yiming Cui, Baoxin Wang, Min Lin, Dayong Wu, and Zhigang Chen. CINO: A Chinese minority pre-trained language model. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 3937–3949, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.346.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130b: An open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-Aw0rrrPUF.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32*, Vancouver, Canada, 2019. URL https://openreview.net/references/pdf?id=S1qBAf6rr.