TeleChat: An Open-source Billingual Large Language Model

Zihan Wang[*], **XinZhang Liu**[*], **Shixuan Liu**[*], **Yitong Yao**[*], **Yuyao Huang**[*],
Mengxiang Li, Zhongjiang He, Yongxiang Li, Luwen Pu, Huinan Xu,
Chao Wang[†], **Shuangyong Song**[†]
Institute of Artificial Intelligence(TeleAI),
China Telecom Corp Ltd
{wangzh54,liuxz,liusx14,yaoyt2,huangyy121,hezj,
liyx25,pulw,xuhn,wangc17,songshy}@chinatelecom.cn

Abstract

In this paper, we present **TeleChat**, a collection of large language models (LLMs) with parameters of 7 billion and 12 billion. TeleChat is initially pretrained on an extensive corpus containing a diverse collection of texts from both English and Chinese languages, encompassing trillions of tokens. Subsequently, the model undergoes fine-tuning to align with human preferences, following a detailed methodology that we describe. We evaluate the performance of TeleChat on various tasks, including general dialogue generation, language understanding, mathematics, reasoning, code generation, and knowledge-based question answering. Our findings indicate that TeleChat achieves state-of-the-art performance to other open-source models of similar size across a wide range of public benchmarks. To support future research and applications utilizing LLMs, we release the fine-tuned model checkpoints of TeleChat-7B and TeleChat-12B, along with code and a portion of our filtered high-quality pretraining data, to the public community[1].

1 Introduction

The research community has witnessed substantial proliferation of open large language models (LLMs). Following the introduction of ChatGPT(OpenAI, 2022), there have been thrilling advancements and applications of LLMs, but the majority of prominent LLMs, such as GPT-4(OpenAI, 2023) and PaLM-2(Anil et al., 2023), are restrictive in their technological sharing. In contrast, a steady stream of openly accessible text-based LLMs has emerged, including OPT(Zhang et al., 2022), BLOOM(Scao et al., 2022), LLAMA(Touvron et al., 2023a), LLAMA 2(Touvron et al., 2023b), etc. Furthermore, there exist various other LLMs that have been designed with a focus on Chinese-English bilingual language generation, including models such as Baichuan-2(Yang et al., 2023), Qwen(Bai et al., 2023), InternLM(InternLM_Team, 2023) and SkyWork(Wei et al., 2023). While these models offer comprehensive details about their pretraining strategies, they often lack transparency in their instruction finetuning processes for chat models, including limited disclosure of the finetuning data composition, methods for concatenating multiturn dialog data, and techniques employed to enhance conversational performance.

To encourage reproducibility of fine-tuned LLMs and foster responsible development of LLMs, we release TeleChat, a collection of pretrained language models and chat models that have been fine-tuned using human alignment techniques including supervised fine-tuning and reinforcement learning. In particular, we provide a comprehensive explanation of our model architecture and the approach we used to extend TeleChat's context window to 96k in Section 2. Furthermore, in Section 3, we delve into the specifics of our pretraining dataset and cleaning techniques. We then discuss alignment with human preferences in Section 4 and 5. Additionally, in Section 6, we conduct a thorough analysis of the model's performance on standard benchmark tasks and general dialogue generation. Throughout the development of TeleChat, we gain insights regarding mitigating hallucination with a knowledge graph, which is discussed in Section 7. Furthermore, we describe our parallel computing method in Section 8. Our contribution are listed as follows:

- We release TeleChat, a suite of pretrained and fine-tuned large language models with parameter sizes of 7 billion and 12 billion. We release model checkpoints and code to the public community.
- We present our comprehensive data cleaning workflow, and release a portion of our high-quality training corpus, comprising more than

[*] These authors contributed equally to this work.
[†] Corresponding Authors.
[1] https://github.com/Tele-AI/Telechat

1TB of text data and exceeding 160 billion tokens. To the best of our knowledge, this marks the largest open Chinese corpus for language model pre-training to the date.

- We disclose a comprehensive description of our supervised fine-tuning methodology, an aspect that is frequently overlooked in reports of other publicly available models. Furthermore, TeleChat stands out with its longest context length among open-source large language models.

2 Model Design

2.1 Model Architecture

TeleChat is an autoregressive transformer model that employs a stack of transformer-decoder layers, whose architecture largely follows that of GPT-3(Brown et al., 2020). However, TeleChat deviates from the original transformer model in several notable ways, drawing inspiration from influential language models such as LLaMA(Touvron et al., 2023a) and BLOOM(Scao et al., 2022). The key parameters of the architecture are summarized in Table 1.

Rotary Position Embedding. We use Rotary Positional Embedding (RoPE(Su et al., 2022)) to encode absolute positions with explicit integration of relative position dependencies. To further optimize computational efficiency and minimize memory usage, we implement Flash Attention v2 in the attention modules(Dao, 2023).

Normalizations. To ensure robust training, we incorporate an additional layer normalization step after the initial embedding layer for TeleChat, drawing inspiration from the methodology employed in BLOOM(Scao et al., 2022). However, we diverge from BLOOM by replacing conventional layer normalization with RMSNorm(Zhang and Sennrich, 2019), which has been shown to enhance the stability and performance of transformer models. Additionally, we adopt pre-normalization in each layer instead of post-normalization, a design choice that has been found to improve the training stability of transformer models.

Activations We utilize the SwiGLU activation function(Shazeer, 2020) in the feed forward network (FFN) of TeleChat, and diminish the FFN feed-forward dimension to less than four times the hidden size, adhering to established conventions in prior research(Touvron et al., 2023a)(Wei et al., 2023).

2.2 Extending Context Window

Large language models (LLMs) often encounter input contexts with a significant number of tokens in different scenarios. Hence, it is crucial for LLMs to have long-range capabilities and efficiently handle context lengths that exceed their initial pre-training limitations.

In our approach, we utilize NTK-aware interpolation techniques (bloc97, 2023) to redistribute the interpolation pressure across multiple dimensions. Additionally, we address performance degradation caused by fluctuations in context length during multiple forward-passes by employing a Dynamic NTK-aware interpolation mechanism, in which the interpolation scaling factor is designed as a continuous variable and is updated according to real-time context length.

To enhance TeleChat's long-context capabilities, we employ Multi-stage Long-context Training during supervised finetuning and attention-Scaling techniques(Peng et al., 2023) during the inference stage. Multi-stage Long-context Training periodically extends the context length during training, while attention-Scaling adjusts the attention mechanism by rescaling the dot product relative to the context-to-training length ratio. This ensures stable attention entropy as the context length increases. For a detailed description of Multi-stage Long-context Training, please refer to section 4.2.3. Experimental results demonstrate that these techniques enable TeleChat to extend its context window to over 96k tokens successfully, which achieves longest context length among open-source large language models.

3 Pretraining Stage

During pretraining stage, we train the model from scratch using a substantial amount of data. In this section, we introduce our data collection and cleaning method (Section 3.1 and 3.2), training details (Section 3.3), and tokenizer (Section 3.4).

3.1 Data Collection

TeleChat's pretraining corpus is curated from a wide range of data sources, including both general-purpose and domain-specific data. The general-purpose data comprises a vast range of sources, such as web pages, social platforms, encyclopedias, books, academic papers, code repositories, and

Models	**layer num**	**attention heads**	**hidden size**	**FFN hidden size**	**vocab size**
TeleChat-7B	30	32	4096	12288	160256
TeleChat-12B	38	32	5120	12288	160256

Table 1: Detailed model architecture parameters for TeleChat's 7B and 12B models.

Datasets	**Percentage%**
web page	22
books	11
community QA	7
social sharing	8
documents and reports	13
paper	2
code repository	12
chat data	13
others	12
Chinese	45
English	35
Code	11
Math	9

Table 2: The distribution of various categories of TeleChat's pretraining data.

more. In terms of domain-specific data, we gather corpus from twenty distinct sectors, including finance, construction, health and social work, aligning with national industry classifications[2]. Furthermore, we consistently gather and accumulate real-time data to ensure comprehensive coverage of the most up-to-date information. During the data collection stage, we acquire diverse and extensive pre-training data on a petabyte scale. The distribution of our pretraining data is displayed in Table 2.

3.2 Data Preprocessing

We devise a comprehensive data cleaning procedure to ensure the quality of our pretraining data. Our data clean procedure consists of rule-based filtering, deduplication, high-quality data selection, and data security filtering.

Rule-based Filtering. Heuristic rules are applied to clean the text efficiently and effectively. For instance, we filter out extremely short or low-information texts, discard texts with excessive or minimal punctuation, and replace HTML tags with natural language. Additionally, we exclude data in languages other than Chinese and English, as well as non-text multimodal data.

Deduplication. Performing global deduplication on a large amount of data is unacceptably slow, therefore we perform a hierarchical deduplication method. First, we eliminate duplicate data from similar sources within groups using URL deduplication, which removes approximately half of the duplicate data. Next, we utilize a 128-bit SimHash algorithm for Document-level Deduplication that removes duplicate articles. Finally, we employ Minhash and Jaccard similarity methods to perform Paragraph-level Deduplication, filtering out a large number of homogeneous advertisements and other heavily redundant texts.

High-quality Selection We utilize a 5-gram Kneser-Ney model, as implemented in the KenLM library(Heafield, 2011), to train on existing high-quality corpora and subsequently compute the perplexity of each paragraph. Instead of simply discard texts with high perplexity, we split the data into three even parts: *head*, *middle*, and *tail* based on the perplexity score. The *head* part will be sampled more frequently, while the *tail* part will be sampled less.

Security Filtering. To ensure the security of our dataset, we employ a multi-model classification approach that identifies and removes pornography, advertising, violent, and politically sensitive content. Moreover, we utilize obfuscation techniques to protect personal privacy data.

3.3 Training Details

Batch Generation. To generate data batches, we employ a process of shuffling and concatenating the corpus obtained from the same source, ensuring consistency in the data. Furthermore, to align with the specified context lengths (e.g., 4096), the data is truncated and concatenated with other data samples.

Training Objectives. The method utilized in the pretraining stage is known as autoregressive language modeling, which involves iteratively predicting the probability of the subsequent token in the sequence. We represent the joint probability of

[2] https://www.stats.gov.cn/english/NewsEvents/200306/t20030619_25521.html

tokens in a text as:

$$p(\mathbf{x}) = p(x_1, \cdots, x_T) = \sum_{t=1}^{T} p(x_t | x_{<t}) \qquad (1)$$

Where $\mathbf{x}$ is a sequence of tokens, and we calculate the probability of each token x_t based on the tokens that come before it, denoted as $x_{<t}$. The model is trained to optimize this probability across the entire training corpus.

Optimizer. We utilize the widely used Adam(Kingma and Ba, 2017) optimizer for pretraining optimization. We employ a cosine learning rate schedule, where the peak learning rate is specified for each model size. The learning rate gradually decays until it reaches a minimum learning rate of 10% of the peak value. The hyperparameters are set as follows: $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-5}$. A weight decay of 10^{-4} is applied to all model parameters except for bias.

Ramp-up Batch. In order to enable the model to converge faster at the very beginning of pretraining, we employ a technique called ramp-up batch size, which involves starting with a small batch size and gradually increasing it linearly to the maximum batch size over a certain number of steps.

Precision.The use of the float16 data type has been recognized as a potential contributing factor to training divergences in LLMs. To address this, we pretrain all models using bfloat16(Wang and Kanwar, 2019), a data type that shares the same dynamic range as float32. Additionally, we employ mixed-precision training, wherein precision-sensitive operations like gradient accumulation, softmax, and weight updating are performed with float32 precision, while the remaining operations are carried out with bfloat16 precision.

The specific hyperparameters are presented in Table 3.

3.4 Tokenizer

We utilize Hugging Face's tokenizers to implement the BBPE algorithm, training the tokenizer on a diverse dataset comprising Chinese, English, code, and mathematical data. This process results in a tokenizer with a vocabulary size of 160,130, which is subsequently padded to 160,256. Additionally, we use special tokens to differentiate dialogue roles and turns, and also incorporate specific designs to mitigate potential injection attacks.

HyperParams	TeleChat-7B	TeleChat-12B
Peak lr	3e-4	1.5e-4
ramp-up batch size	288/72/1,500,000	240/80/2,000,000
batch size	16M	16M
warm up fraction	0.01	0.01
# training tokens	1.0T	1.2T

Table 3: The hyperparameter details utilized during the pretraining stage of TeleChat's 7B and 12B variants. The ramp-up batch size is expressed in the format of <start batch size >/<batch size increment>/<ramp-up samples>. For example, 240/80/2,000,000 indicates that the training begins with a batch size of 240 and increments by 80 for each time. The total ramp-up phase encompasses 2,000,000 samples.

4 Supervised Fine-Tuning Stage

We employ supervised fine-tuning (SFT) stage after the pretraining stage to effectively accomplish various real-world tasks. In this section, we provide detailed information about our data collection and annotation method in Section 4.1, followed by an in-depth discussion of our methodology and experimental details in Section 4.2 and Section 4.3.

4.1 Human Data Collection

We brought together a team of annotators to carry out the manual data annotation process. Our annotators are all native Chinese speakers, boasting a range of academic backgrounds including Computer Science, Law, Chinese language and literature, and other related fields. We ask the human annotators to label varied prompts and organize them into conversations, harnessing our annotation platform for efficient and high-quality annotations. We work closely with the labelers, providing them with clear instructions for each task and addressing their questions promptly.

We collect over 100,000 supervised fine-tuning samples using the aforementioned annotation strategies and train our model accordingly. The statistics of the top 30 categories in our supervised-finetuning data is displayed in Supplement Material Section A.

4.2 Training Methodology

In this section, we present a comprehensive explanation of our training approach during the supervised fine-tuning stage, an aspect that is frequently overlooked in reports of other open-sourced models.

4.2.1 Data Organization

Our dataset spans various domains, such as General Q&A, creative writing, machine translation, code generation, math & reasoning, and more. To ensure that each domain is represented appropriately, we assign respective resampling weights to each dataset based on their importance. Then, we sample single-round and multi-round conversations from each dataset using their corresponding resampling weights. The sampled conversations are then shuffled and concatenated, followed by pre-padding them to a predetermined length (e.g., 4096 or 8192) to ensure consistent input length. We use special tokens `<_user>`, `<_bot>`, and `<_end>` to denote the beginning of a question, the start of an answer, and the end of an answer respectively. To ensure diversity in the combination of data, the datasets are resampled and re-shuffled for each training epoch. We fine-tuned the model in a supervised manner based on this carefully curated instruction dataset.

4.2.2 Noisy Embedding Fine Tuning

In this section, we introduce our method for enhancing the answer quality of large language models (LLMs) through noisy embedding fine-tuning (`NEFTUNE`), inspired by the work of(Jain et al., 2023). Specifically, `NEFTune` modifies the input embeddings by adding a random noise vector to them. The noise is generated by sampling independent and identically distributed (i.i.d) uniform entries, each in the range $[-1, 1]$, and then scaling the entire noise vector by a factor of $\alpha/\sqrt{Ld}$, where L is the sequence length, d is the embedding dimension, and α is a tunable hyperparameter.

We observe that while `NEFTune` enhance the model's performance in scenarios with limited training data, its benefits diminish as the size of the training dataset increases. This is likely due to the model's reduced tendency to overfit on larger datasets. To investigate this further, we conduct experiments using TeleChat-7B fine-tuned models with and without the implementation of `NEFTune`. Our findings reveal that when the model is trained on the 10,000 samples, `NEFTune` achieves a 55% win rate against its counterpart without `NEFTune`, as determined by human evaluators. Some examples are shown in Supplement Material Section B. However, when the model is trained on the entire dataset consisting of 40,000 samples, `NEFTune` loses its advantage, resulting in only a 48% win rate against its counterpart without `NEFTune`.

4.2.3 Multi-stage Long-context Training.

During the supervised fine-tuning stage, we gradually increase the training length, enabling the model to activate and strengthen its ability to understand extensive dependencies while preserving its foundational skills. Specifically, we initiate the training with a sequence length of 8,192, building upon the foundation model trained on a sequence length of 4,096. At the 3/4 mark of the training procedure, we transit to a training sequence length of 16,384. Note that we employ the ntk-aware extrapolation method when working with sequence lengths of 8,192 and 16,384. This approach helps us mitigate the difficulties encountered during the transition, allowing for a smooth adjustment in the training sequence length for the model. Training details for TeleChat-7B's multi-stage long-context training is shown in Table 4, and experiment results is displayed in Table 5

4.3 Training Details

Similar to the pretraining phase, we employ next-token prediction as the training task. However, we introduce loss masks for system information and user input questions to ensure that the loss is exclusively calculated for the output answer.

The model undergoes a total of 40,000 steps, with the first 30,000 steps involving training with a sequence length of 8,192, and the remaining 10,000 steps involving training with a sequence length of 16,384, as illustrated in section 4.2.3. In the training process, we utilize the same optimizer as in the pretraining stage, as described in section 3.3.

5 Reinforcement Learning

In this section, we introduce reinforcement learning to align chat models with human preference, aiming to make model outputs consistent with safety and norms.

5.1 Reward Model

When collecting prompts of reward dataset, a consensus is that high-quality and diverse prompts are conducive to the training stage of reinforcement learning.

We collect a large number of prompts, including data from both human annotation and internal user testing phases. The final prompt dataset consists of a total of 300 categories. To further get the high quality prompts, we use clustering and centroid selection to select representative prompts.

sequence length	training steps	peak lr	batch size	tensor parallel	pipeline parallel
8,192	30,000	3e-5	8M	2	4
16,384	10,000	4e-5	8M	2	8

Table 4: Training details for TeleChat-7B's multi-stage long-context training. Note that training with a sequence length of 16,384 demands significantly more GPU memory compared to training with 8,192. As a result, it is necessary to increase the pipeline parallel size to 8, and requires 2 nodes to train.

Method	sequence length						
	2048	4096	8192	16384	32768	65536	98304
baseline	4.8122	4.6562	39.3099	98.3102	155.2708	487.3398	447.6295
NTK-aware (8k)	4.8122	4.6562	5.1904	4.7155	8.6351	77.7478	79.9256
NTK-aware+logN (8k)	4.8122	4.6562	5.1904	4.0353	4.1408	9.4080	7.9711
NTK-aware (16k)	7.6916	7.9900	7.9580	5.1217	4.7932	10.5444	10.3614
NTK-aware+logN (16k)	7.6916	7.9900	7.9580	5.1217	4.7195	8.9751	7.6822

Table 5: Our experiments with TeleChat-7B's long-context inferences illustrate the effectiveness of employing techniques such as NTK-aware extrapolation, attention scaling, and multi-stage long-context training. These approaches result in a significant reduction in perplexity as the context length increases and enable our model to achieve a low perplexity when extrapolating to 96K tokens.

All prompts are firstly convert to embeddings using bge-large-zh [3]. Then we employ elbow clustering algorithms within each categories that aims to find the ideal number of clusters. The closest prompt to each cluster centroid will be selected. In addition, we randomly sampled the prompts in the cluster (except the closest prompt) to ensure the diversity of reward dataset, while the remain is used for reinforcement learning. The responses are collected from TeleChat models of different training stages and reasoning strategies, allowing sampling rich responses for annotation.

Moreover, for improving the accuracy and reducing the difficulty of annotations, we simplify the task of ranking responses with human annotation. A straightforward classification task is introduced, where responses can be categorized under three distinct labels: good, medium, and bad. The basic criteria of this assessment includes but is not limited to safety, factuality, fluency, normality, etc. By evaluating the responses through these aspects, annotators can rank responses consistently. The responses between each pair of distinct labels under the same prompt can be combined with each other to form ranked pairs for subsequent training.

During the training stage, we use the same training objectives as LLaMA2, adding margin in the loss function to teach the reward model to assign more difference scores to response pairs with more difference. The training data distribution, adding margin size and test accuracy of Reward Model on three types of data pairs are shown in Table 6.

5.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is widely used for LLM alignment and its mechanism is collaboratively working including four models: actor model, critic model, reference model and reward model. From the experience of (Yang et al., 2023) and (Bai et al., 2023), the critic model updates 50 steps firstly before actor model. The KL divergence coefficient is setting to 0.1 and apply a normalization process to the rewards, which accounts for the moving average. The learning rates for our actor and critic models are configured at $5 \times 10 - 6$ and $3 \times 10 - 6$ respectively through experiments. We get the chat model eventually after training for 400 steps.

6 Experiment

6.1 Evaluation on Standard Benchmarks

In this chapter, we evaluate the zero-shot and few-shot capabilities of TeleChat from various perspectives using standard benchmarks. We select a list of open source models as baselines, including LLaMA 2-Chat (7B, 13B), InternLM-Chat (7B), Belle-LLaMA-2 (13B), Baichuan 2 (7B, 13B), ChatGLM 2-6B, ChatGLM 3-6B, Qwen-Chat (7B, 14B).

[3] `https://huggingface.co/BAAI/bge-large-zh-v1.5`

Type of data	**good & bad**	**medium & bad**	**good & medium**
Distribution	18.2%	21.1%	65.7%
Margin	1	2/3	1/3
Test Accuracy	70.1%	66.0%	86.4%

Table 6: Training data distribution, adding margin and test accuracy of Reward Model on different type of data pairs.

Model	MMLU (5-shot)	C-Eval (5-shot)	CMMLU (5-shot)	AGIEval (zero-shot)	GAOKAO (zero-shot)	CSL (zero-shot)	CHID (zero-shot)	EPRSTMT (zero-shot)	GSM8K (4-shot)	MATH (4-shot)	HumanEval (zero-shot)
LLaMA2-7B-chat	46.2	31.9	31.5	28.5	16.1	58.8	44.1	57.5	26.3	3.9	12.2
LLaMA2-13B-chat	54.6	36.2	38.7	32.3	18.6	61.2	48	59.4	29.6	5.0	18.9
ChatGLM2-6B-chat	45.9	52.6	49.3	39	46.4	61.2	57.9	71.2	28.8	6.5	11
ChatGLM3-6B-chat	51.9	53.8	54	38.9	49.3	65.6	63.4	85	56.7	18.7	61
InternLM-7B-chat	52	54.1	52.6	43.7	45.8	70	79.7	88.8	34.6	5.6	12.8
Baichuan2-7B-chat	52.8	55.6	54	35.3	39.7	60	75.2	87.5	32.8	6	13.4
Baichuan2-13B-chat	57	56.7	58.4	40	51.4	63.1	78.2	87.5	55.3	8.6	17.7
Qwen-7B-chat	56.6	59.3	59.5	41.3	63.3	63.1	72.3	88.8	52.5	10.3	26.2
Qwen-14B-chat	66.4	71.7	70.0	47.3	76.5	55.6	72.3	91.2	61.0	26.8	36.6
TeleChat-7B-chat	54.4	63.1	64.3	46.8	57.7	66.8	88.0	87.5	36.7	10.3	14.6
TeleChat-12B-chat	73.3	66.6	74.2	51.7	53.1	60.6	83.2	86.3	57.2	16.0	22.0

Table 7: Results of TeleChat compared with other large language models on eleven general benchmarks.

Model	**General Q&A**	**Safety Task**	**Hallucination Task**
GPT3.5	66.3	73.9	**72.2**
Qwen(14B)	66.4	70.7	64.2
BaiChuan2(7B)	59.1	71.9	40.2
TeleChat-12B	**71.4**	**75.4**	66.2

Table 8: The evaluation results of TeleChat and other models on general dialogue Q&A, safety task and hallucination task. The best results are shown in **bold**.

6.1.1 Overall Performance

We evaluate TeleChat on multiple challenging benchmarks. The detailed information of test benchmarks is as follows:

- MMLU(Hendrycks et al., 2021a): An English benchmark covering 57 tasks, which are mostly college level.
- CMMLU(Li et al., 2023): A Chinese benchmark to evaluate a LLM's knowledge and reasoning ability.
- C-Eval(Huang et al., 2023): A comprehensive Chinese benchmark, containing more than 10 thousands questions and four difficulty levels.
- GAOKAO-Bench(Zhang et al., 2023): A Chinese evaluation benchmark utilizing Chinese college entrance examination questions.
- AGIEVAL(Zhong et al., 2023): A bilingual evaluation dataset encompassing standardized test questions.
- CSL(Li et al., 2022): A dataset containing multiple Chinese papers, which requires to checks the match between Chinese academic abstracts and their keywords.
- EPRSTMT(Xu et al., 2021): EPRSTMT is a sentiment analysis datasets based on comments on e-commerce websites.
- CHID(Zheng et al., 2019): A reading comprehension benchmark, which requires the model to select the most appropriate idiom to fill in the blanks within the text.
- GSM8K(Cobbe et al., 2021): GSM8K is a dataset of 8.5K high-quality, linguistically diverse, human-written elementary math problems.
- Math(Hendrycks et al., 2021b): A dataset containing 12.5K challenging competition math problems.
- HumanEval(Chen et al., 2021): A code test dataset provided by OpenAI, which consists of 164 programming questions that measure the correctness of code.

We record the detailed experiment results in Table 7. To standardize the evaluation method, we employ the assessment technique provided by OpenCompass to obtain the results on most of the benchmarks. The referenced model results all originate from the open leaderboard of OpenCompass. We observe that TeleChat exhibits superior performance compared to models of the same size. Particularly in terms of the results on the

MMLU, AGIEVAL, CMMLU and CHID datasets, TeleChat's performance surpasses that of other models of equivalent size.

6.2 Evaluation on General Dialogue Tasks

We assess TeleChat's ability to deliver helpful, truthful, and secure responses to user input, using a specific set of prompts that are distinct from our training data. Our test data is categorized into general dialogue generation tasks, safety tasks, and hallucination tasks. We compare TeleChat's output with other models, using GPT-4 as an automatic referee, and then ask human labelers to review and revise the results of GPT-4. The human evaluation process is conducted in a blind manner. Examples of our evaluation dataset is shown in Supplement Material Section C.

The results, presented in Table 8, demonstrate that TeleChat-12B achieves a 99.3% performance level compared to GPT3.5 and outperforms other opensource models of similar sizes. We also showcase TeleChat's capability to address real-world inquiries in Supplement Material Section D.

7 Alleviating Hallucination with Knowledge Graph

Hallucination problems are frequently observed in LLMs, where there is a tendency to generate text that appears coherent and meaningful but lacks real-world existence. In this paper, we address the first type of hallucinations by utilizing structured information representation provided by Knowledge Graphs (KG).

When a query comes, candidate entities are firstly retrieved based on n-gram similarity with query. Subsequently, a random walk of n steps is conducted within the graph, starting from these candidate entities. Finally, all paths obtained through the random walk are sorted based on their relevance to the user's query. The top-k paths are then returned as the final result of the knowledge graph retrieval process. By combining this retrieved knowledge with a prompt, the large language model can process the augmented query, taking into consideration the background knowledge provided by the knowledge graph. We evaluated the TeleChat's ability to answer factual questions in the China Conference on Knowledge Graph and Semantic Computing (CCKS) 2020 Knowledge Graph based Q&A task[4]. Without the introduction of the knowledge graph, the accuracy of TeleChat on this task is recorded at 0.19. However, after incorporating the relevant knowledge by adding the top 10 relevant paths from the knowledge graph, the accuracy significantly improves to 0.69. This demonstrates the effectiveness of integrating the knowledge graph in enhancing the TeleChat's ability to provide accurate answers to factual questions.

[4]https://sigkg.cn/ccks2020/?page_id=69

8 Engineering

8.1 Hardware

TeleChat is trained on a total of 80 nodes, each having 8 Nvidia A100 Sxm 40GB GPUs. Each node is equipped with 2x Intel 6348 (28 Cores, 2.60 GHz) CPUs, 8x NVLink A100 GPUs, 512GB of RAM, and a 2GB cache RAID card. All nodes are interconnected using InfiniBand (IB) for networking. To enhance data transmission speed and mitigate bandwidth constraints, we employ NVIDIA's GPUDirect RDMA (GRDMA) and utilize the Scalable Hierarchical Aggregation and Reduction Protocol (SHARP).

8.2 Parallel Computing

TeleChat is trained using the Megatron-DeepSpeed framework (Smith et al., 2022) for large-scale distributed training. TeleChat successfully leverages 3D parallelism, which integrates tensor parallelism, pipeline parallelism, and data parallelism to enable efficient distributed training. We scale our system to utilize hundreds of GPUs with extensive GPU utilization, achieving a peak performance of 180 TFLOPs using A100 GPUs, which accounts for 57.6% of the theoretical peak performance of 312 TFLOPs.

9 Conclusions

In this paper, we introduced TeleChat, a collection of large language models (LLMs) with 7 billion and 12 billion parameters. We detailed the pretraining process, supervised fine-tuning, reinforcement learning, and the integration of a knowledge graph to enhance the model's performance. We evaluated TeleChat on various benchmarks and compared its performance with other open-source models, TeleChat demonstrates superior performance in general dialogue tasks, knowledge-based question answering, and various other benchmarks, showcasing its potential for diverse real-world applications. We release model checkpoints, code, and a portion of our filtered high-quality pretraining data totaling 160 billion tokens to the public community.

Limitations

While TeleChat demonstrates impressive performance across various language tasks, there are several limitations to consider. Firstly, the extensive computational resources required for training and inference may also pose challenges for wider adoption and accessibility. Additionally, the integration of knowledge graphs, while effective in reducing hallucination, may introduce biases or inaccuracies if the underlying knowledge graph data is incomplete or outdated. Furthermore, the evaluation of TeleChat's performance, while comprehensive, may not fully capture its real-world applicability and potential limitations in specific domains or scenarios. Addressing these limitations will be crucial for the responsible and ethical deployment of TeleChat in real-world applications.

Ethics Statement

The development and evaluation of TeleChat prioritize ethical considerations. We prioritize privacy, consent, and fairness in data usage, and have made model checkpoints, code, and a portion of the training data publicly available for transparency and reproducibility. We are committed to addressing ethical concerns such as bias, privacy, and misinformation, and will continue to monitor and improve TeleChat's behavior in alignment with societal values.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

bloc97. 2023. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N.

Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

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. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*.

InternLM_Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities.

Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Neftune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023. Cmmlu: Measuring massive multitask language understanding in chinese.

Yudong Li, Yuqing Zhang, Zhe Zhao, Linlin Shen, Weijie Liu, Weiquan Mao, and Hui Zhang. 2022. Csl: A large-scale chinese scientific literature dataset. *arXiv preprint arXiv:2209.05034*.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Teven Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander Rush, Stella Biderman, Albert Webson, Pawan Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Moral, and Thomas Wolf. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Shaden Smith, Md. Mostofa Ali Patwary, Brandon Norick, Patrick Legresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zhang, Rewon Child, Reza Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2022. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

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. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Shibo Wang and Pankaj Kanwar. 2019. Bfloat16: The secret to high performance on cloud tpus, 2019.

Tianwen Wei, Liang Zhao, Lichang Zhang, Bo Zhu, Lijie Wang, Haihua Yang, Biye Li, Cheng Cheng, Weiwei Lü, Rui Hu, Chenxia Li, Liu Yang, Xilin Luo, Xuejie Wu, Lunan Liu, Wenjun Cheng, Peng Cheng, Jianhao Zhang, Xiaoyu Zhang, Lei Lin, Xiaokun Wang, Yutuan Ma, Chuanhai Dong, Yanqi Sun, Yifu Chen, Yongyi Peng, Xiaojuan Liang, Shuicheng Yan, Han Fang, and Yahui Zhou. 2023. Skywork: A more open bilingual foundation model. *arXiv preprint arXiv:2310.19341*.

Liang Xu, Xiaojing Lu, Chenyang Yuan, Xuanwei Zhang, Huilin Xu, Hu Yuan, Guoao Wei, Xiang Pan, Xin Tian, Libo Qin, et al. 2021. Fewclue: A chinese few-shot learning evaluation benchmark. *arXiv preprint arXiv:2107.07498*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *arXiv preprint arXiv:1910.07467*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark.

Chujie Zheng, Minlie Huang, and Aixin Sun. 2019. ChID: A large-scale Chinese IDiom dataset for cloze test. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 778–787, Florence, Italy. Association for Computational Linguistics.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models.