QWEN TECHNICAL REPORT

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, Tianhang Zhu.

Qwen Team, Alibaba Group*

ABSTRACT

Large language models (LLMs) have revolutionized the field of artificial intelligence, enabling natural language processing tasks that were previously thought to be exclusive to humans. In this work, we introduce QWEN[1], the first installment of our large language model series. QWEN is a comprehensive language model series that encompasses distinct models with varying parameter counts. It includes QWEN, the base pretrained language models, and QWEN-CHAT, the chat models finetuned with human alignment techniques. The base language models consistently demonstrate superior performance across a multitude of downstream tasks, and the chat models, particularly those trained using Reinforcement Learning from Human Feedback (RLHF), are highly competitive. The chat models possess advanced tool-use and planning capabilities for creating agent applications, showcasing impressive performance even when compared to bigger models on complex tasks like utilizing a code interpreter. Furthermore, we have developed coding-specialized models, CODE-QWEN and CODE-QWEN-CHAT, as well as mathematics-focused models, MATH-QWEN-CHAT, which are built upon base language models. These models demonstrate significantly improved performance in comparison with open-source models, and slightly fall behind the proprietary models.

*Authors are ordered alphabetically by the last name. Correspondence to: ericzhou.zc@alibaba-inc.com.

[1]QWEN is a moniker of Qianwen, which means "thousands of prompts" in Chinese. The pronunciation of "QWEN" can vary depending on the context and the individual speaking it. Here is one possible way to pronounce it: /kwɛn/.

Contents

1 INTRODUCTION

Large language models (LLMs) (Radford et al., 2018; Devlin et al., 2018; Raffel et al., 2020; Brown et al., 2020; OpenAI, 2023; Chowdhery et al., 2022; Anil et al., 2023; Thoppilan et al., 2022; Touvron et al., 2023a;b) have revolutionized the field of artificial intelligence (AI) by providing a powerful foundation for complex reasoning and problem-solving tasks. These models have the ability to compress vast knowledge into neural networks, making them incredibly versatile agents. With a chat interface, LLMs can perform tasks that were previously thought to be the exclusive domain of humans, especially those involving creativity and expertise (OpenAI, 2022; Ouyang et al., 2022; Anil et al., 2023; Google, 2023; Anthropic, 2023a;b). They can engage in natural language conversations with humans, answering questions, providing information, and even generating creative content such as stories, poems, and music. This has led to the development of a wide range of applications, from chatbots and virtual assistants to language translation and summarization tools.

LLMs are not just limited to language tasks. They can also function as a generalist agent (Reed et al., 2022; Bai et al., 2022a; Wang et al., 2023a; AutoGPT, 2023; Hong et al., 2023), collaborating with external systems, tools, and models to achieve the objectives set by humans. For example, LLMs can understand multimodal instructions (OpenAI, 2023; Bai et al., 2023; Liu et al., 2023a; Ye et al., 2023; Dai et al., 2023; Peng et al., 2023b), execute code (Chen et al., 2021a; Zheng et al., 2023; Li et al., 2023d), use tools (Schick et al., 2023; LangChain, Inc., 2023; AutoGPT, 2023), and more. This opens up a whole new world of possibilities for AI applications, from autonomous vehicles and robotics to healthcare and finance. As these models continue to evolve and improve, we can expect to see even more innovative and exciting applications in the years to come. Whether it's helping us solve complex problems, creating new forms of entertainment, or transforming the way we live and work, LLMs are poised to play a central role in shaping the future of AI.

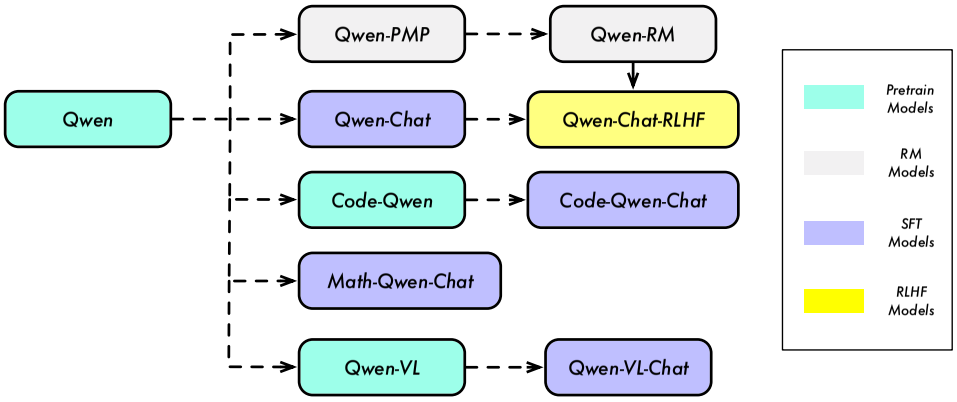

Figure 1: **Model Lineage of the Qwen Series.** We have pretrained the language models, namely QWEN, on massive datasets containing trillions of tokens. We then use SFT and RLHF to align QWEN to human preference and thus we have QWEN-CHAT and specifically its improved version QWEN-CHAT-RLHF. Additionally, we also develop specialized models for coding and mathematics, such as CODE-QWEN, CODE-QWEN-CHAT, and MATH-QWEN-CHAT based on QWEN with similar techniques. Note that we previously released the multimodal LLM, QWEN-VL and QWEN-VL-CHAT (Bai et al., 2023), which are also based on our QWEN base models.

Despite their impressive capabilities, LLMs are often criticized for their lack of reproducibility, steerability, and accessibility to service providers. In this work, we are pleased to present and release the initial version of our LLM series, QWEN. QWEN is a moniker that derives from the Chinese phrase Qianwen, which translates to "thousands of prompts" and conveys the notion of embracing a wide range of inquiries. QWEN is a comprehensive language model series that encompasses distinct models with varying parameter counts. The model series include the base pretrained language models, chat models finetuned with human alignment techniques, i.e., supervised finetuning (SFT), reinforcement learning with human feedback (RLHF), etc., as well as specialized models in coding and math. The details are outlined below:

1. The base language models, namely QWEN, have undergone extensive training using up to 3 trillion tokens of diverse texts and codes, encompassing a wide range of areas. These models have consistently demonstrated superior performance across a multitude of downstream tasks, even when compared to their more significantly larger counterparts.
2. The QWEN-CHAT models have been carefully finetuned on a curated dataset relevant to task performing, chat, tool use, agent, safety, etc. The benchmark evaluation demonstrates that the SFT models can achieve superior performance. Furthermore, we have trained reward models to mimic human preference and applied them in RLHF for chat models that can produce responses preferred by humans. Through the human evaluation of a challenging test, we find that QWEN-CHAT models trained with RLHF are highly competitive, still falling behind GPT-4 on our benchmark.
3. In addition, we present specialized models called CODE-QWEN, which includes CODE-QWEN-7B and CODE-QWEN-14B, as well as their chat models, CODE-QWEN-14B-CHAT and CODE-QWEN-7B-CHAT. Specifically, CODE-QWEN has been pre-trained on extensive datasets of code and further fine-tuned to handle conversations related to code generation, debugging, and interpretation. The results of experiments conducted on benchmark datasets, such as HumanEval (Chen et al., 2021b), MBPP (Austin et al., 2021), and HumanEvalPack (Muennighoff et al., 2023), demonstrate the high level of proficiency of CODE-QWEN in code understanding and generation.
4. This research additionally introduces MATH-QWEN-CHAT specifically designed to tackle mathematical problems. Our results show that both MATH-QWEN-7B-CHAT and MATH-QWEN-14B-CHAT outperform open-sourced models in the same sizes with large margins and are approaching GPT-3.5 on math-related benchmark datasets such as GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).
5. Besides, we have open-sourced QWEN-VL and QWEN-VL-CHAT, which have the versatile ability to comprehend visual and language instructions. These models outperform the current open-source vision-language models across various evaluation benchmarks and support text recognition and visual grounding in both Chinese and English languages. Moreover, these models enable multi-image conversations and storytelling. Further details can be found in Bai et al. (2023).

Now, we officially open-source the 14B-parameter and 7B-parameter base pretrained models QWEN and aligned chat models QWEN-CHAT[2]. This release aims at providing more comprehensive and powerful LLMs at developer- or application-friendly scales.

The structure of this report is as follows: Section 2 describes our approach to pretraining and results of QWEN. Section 3 covers our methodology for alignment and reports the results of both automatic evaluation and human evaluation. Additionally, this section describes details about our efforts in building chat models capable of tool use, code interpreter, and agent. In Sections 4 and 5, we delve into specialized models of coding and math and their performance. Section 6 provides an overview of relevant related work, and Section 7 concludes this paper and points out our future work.

2 PRETRAINING

The pretraining stage involves learning vast amount of data to acquire a comprehensive understanding of the world and its various complexities. This includes not only basic language capabilities but also advanced skills such as arithmetic, coding, and logical reasoning. In this section, we introduce the data, the model design and scaling, as well as the comprehensive evaluation results on benchmark datasets.

2.1 DATA

The size of data has proven to be a crucial factor in developing a robust large language model, as highlighted in previous research (Hoffmann et al., 2022; Touvron et al., 2023b). To create an effective pretraining dataset, it is essential to ensure that the data are diverse and cover a wide range

[2]GitHub: `https://github.com/QwenLM/Qwen`.

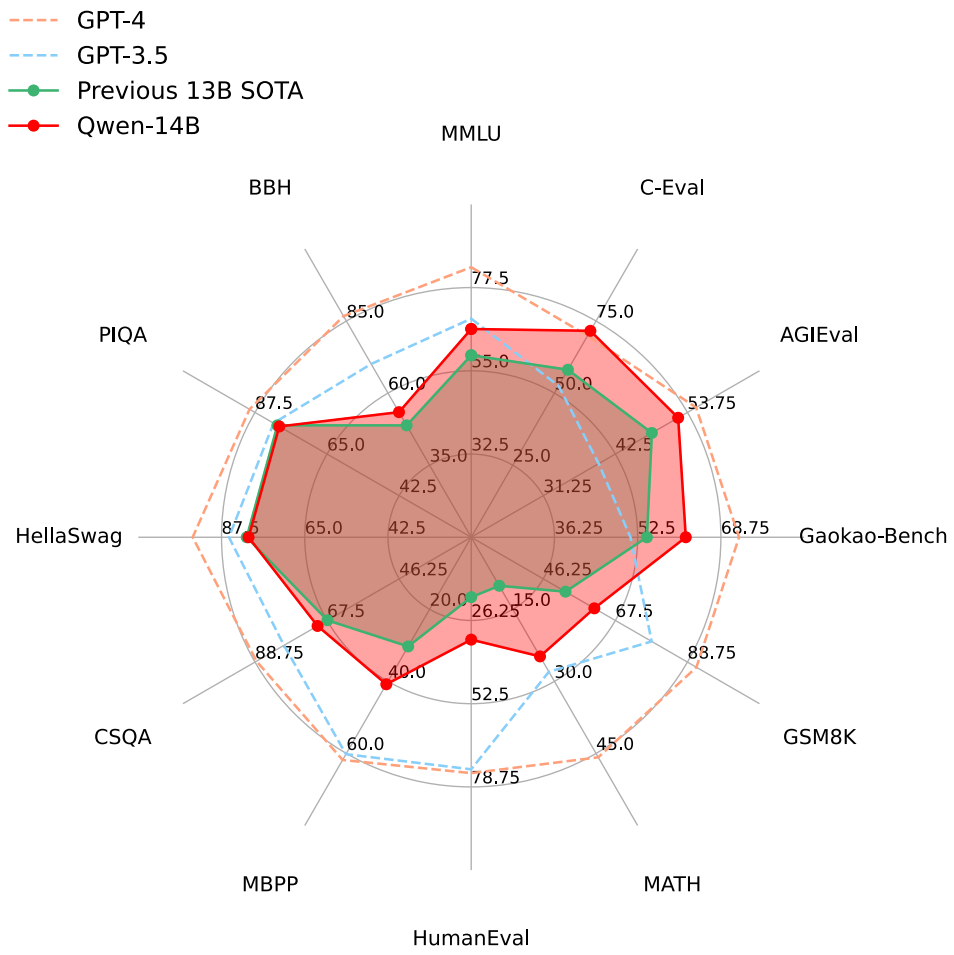

Figure 2: **Performance of GPT-4, GPT-3.5, the previous 13B SOTA, as well as QWEN-14B.** We demonstrate the results on 12 datasets covering multiple domains, including language understanding, knowledge, reasoning, etc. QWEN significantly outperforms the previous SOTA of similar model sizes, but still lag behind both GPT-3.5 and GPT-4.

of types, domains, and tasks. Our dataset is designed to meet these requirements and includes public web documents, encyclopedia, books, codes, etc. Additionally, our dataset is multilingual, with a significant portion of the data being in English and Chinese.

To ensure the quality of our pretraining data, we have developed a comprehensive data preprocessing procedure. For public web data, we extract text from HTML and use language identification tools to determine the language. To increase the diversity of our data, we employ deduplication techniques, including exact-match deduplication after normalization and fuzzy deduplication using MinHash and LSH algorithms. To filter out low-quality data, we employ a combination of rule-based and machine-learning-based methods. Specifically, we use multiple models to score the content, including language models, text-quality scoring models, and models for identifying potentially offensive or inappropriate content. We also manually sample texts from various sources and review them to ensure their quality. To further enhance the quality of our data, we selectively up-sample data from certain sources, to ensure that our models are trained on a diverse range of high-quality content. Finally, we have built a dataset of up to 3 trillion tokens.

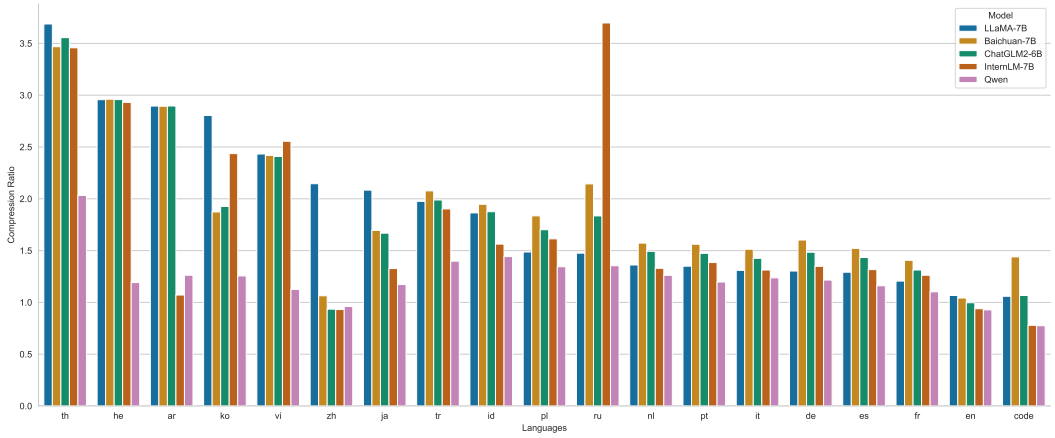

Figure 3: **Encoding compression rates of different models.** We randomly selected 1 million document corpora of each language to test and compare the encoding compression rates of different models (with XLM-R (Conneau et al., 2019), which supports 100 languages, as the base value 1, not shown in the figure). As can be seen, while ensuring the efficient decoding of Chinese, English, and code, QWEN also achieves a high compression rate for many other languages (such as th, he, ar, ko, vi, ja, tr, id, pl, ru, nl, pt, it, de, es, fr, etc.), equipping the model with strong scalability as well as high training and inference efficiency in these languages.

2.2 TOKENIZATION

The design of vocabulary significantly impacts the training efficiency and the downstream task performance. In this study, we utilize byte pair encoding (BPE) as our tokenization method, following GPT-3.5 and GPT-4. We start with the open-source fast BPE tokenizer, tiktoken (Jain, 2022), and select the vocabulary cl100k base as our starting point. To enhance the performance of our model on multilingual downstream tasks, particularly in Chinese, we augment the vocabulary with commonly used Chinese characters and words, as well as those in other languages. Also, following Touvron et al. (2023a;b), we have split numbers into single digits. The final vocabulary size is approximately 152K.

The performance of the QWEN tokenizer in terms of compression is depicted in Figure 3. In this comparison, we have evaluated QWEN against several other tokenizers, including XLM-R (Conneau et al., 2019), LLaMA Touvron et al. (2023a), Baichuan Inc. (2023a), and InternLM InternLM Team (2023). Our findings reveal that QWEN achieves higher compression efficiency than its competitors in most languages. This implies that the cost of serving can be significantly reduced since a smaller number of tokens from QWEN can convey more information than its competitors. Furthermore, we have conducted preliminary experiments to ensure that scaling the vocabulary size of QWEN does not negatively impact the downstream performance of the pretrained model. Despite the increase in vocabulary size, our experiments have shown that QWEN maintains its performance levels in downstream evaluation.

2.3 MODEL

2.3.1 ARCHITECTURE

QWEN is designed using a modified version of the Transformer architecture. Specifically, we have adopted the recent open-source approach of training large language models, LLaMA (Touvron et al., 2023a), which is widely regarded as the top open-source LLM. Our modifications to the architecture include:

- **Embedding and output projection**. Based on preliminary experimental findings, we have opted for the untied embedding approach instead of tying the weights of input embedding and output projection. This decision was made in order to achieve better performance with the price of memory costs.

Table 1: **Model sizes, architectures, and optimization hyper-parameters.**

# of Params	Hidden size	Heads	Layers	Learning rate	Batch size	Training tokens
1.8B	2048	16	24	3.0×10^{-4}	4M	2.2T
7B	4096	32	32	3.0×10^{-4}	4M	2.4T
14B	5120	40	40	3.0×10^{-4}	4M	3.0T

- **Positional embedding**. We have chosen RoPE (Rotary Positional Embedding) (Su et al., 2021) as our preferred option for incorporating positional information into our model. RoPE has been widely adopted and has demonstrated success in contemporary large language models, notably PaLM (Chowdhery et al., 2022; Anil et al., 2023) and LLaMA (Touvron et al., 2023a;b). In particular, we have opted to use FP32 precision for the inverse frequency matrix, rather than BF16 or FP16, in order to prioritize model performance and achieve higher accuracy.
- **Bias**. For most layers, we remove biases following Chowdhery et al. (2022), but we add biases in the QKV layer of attention to enhance the extrapolation ability of the model (Su, 2023b).
- **Pre-Norm & RMSNorm**. In modern Transformer models, pre-normalization is the most widely used approach, which has been shown to improve training stability compared to post-normalization. Recent research has suggested alternative methods for better training stability, which we plan to explore in future versions of our model. Additionally, we have replaced the traditional layer normalization technique described in (Ba et al., 2016) with RMSNorm (Jiang et al., 2023). This change has resulted in equivalent performance while also improving efficiency.
- **Activation function**. We have selected SwiGLU (Shazeer, 2020) as our activation function, a combination of Swish (Ramachandran et al., 2017) and Gated Linear Unit (Dauphin et al., 2017). Our initial experiments have shown that activation functions based on GLU generally outperform other baseline options, such as GeLU (Hendrycks & Gimpel, 2016). As is common practice in previous research, we have reduced the dimension of the feed-forward network (FFN) from 4 times the hidden size to $\frac{8}{3}$ of the hidden size.

2.3.2 Context Length Extension

Transformer models have a significant limitation in terms of the context length for their attention mechanism. As the context length increases, the quadratic-complexity computation leads to a drastic increase in both computation and memory costs. In this work, we have implemented simple training-free techniques that are solely applied during inference to extend the context length of the model. One of the key techniques we have used is NTK-aware interpolation (bloc97, 2023), which adjusts the scale to prevent the loss of high-frequency information in a training-free manner. To further improve performance, we have also implemented a trivial extension called dynamic NTK-aware interpolation, which is later formally discussed in (Peng et al., 2023a). It dynamically changes the scale by chunks, avoiding severe performance degradation. These techniques allow us to effectively extend the context length of Transformer models without compromising their computational efficiency or accuracy.

Qwen additionally incorporates two attention mechanisms: LogN-Scaling (Chiang & Cholak, 2022; Su, 2023a) and window attention (Beltagy et al., 2020). LogN-Scaling rescales the dot product of the query and value by a factor that depends on the ratio of the context length to the training length, ensuring that the entropy of the attention value remains stable as the context length grows. Window attention restricts the attention to a limited context window, preventing the model from attending to tokens that are too far away.

We also observed that the long-context modeling ability of our model varies across layers, with lower layers being more sensitive in context length extension compared to the higher layers. To leverage this observation, we assign different window sizes to each layer, using shorter windows for lower layers and longer windows for higher layers.

Table 2: **Overall performance on widely-used benchmarks compared to open-source base models**. Our largest QWEN model with 14 billion parameters outperforms previous 13B SoTA models on all datasets.

Model	Params	MMLU 5-shot	C-Eval 5-shot	GSM8K 8-shot	MATH 4-shot	HumanEval 0-shot	MBPP 3-shot	BBH 3-shot
MPT	7B	30.8	23.5	9.1	3.0	18.3	22.8	35.6
	30B	47.9	-	15.2	3.1	25.0	32.8	38.0
Falcon	7B	27.8	-	6.8	2.3	-	11.2	28.0
	40B	57.0	-	19.6	5.5	-	29.8	37.1
ChatGLM2	6B	47.9	51.7	32.4	6.5	-	-	33.7
InternLM	7B	51.0	53.4	31.2	6.3	10.4	14.0	37.0
	20B	62.1	58.8	52.6	7.9	25.6	35.6	52.5
Baichuan2	7B	54.7	56.3	24.6	5.6	18.3	24.2	41.6
	13B	59.5	59.0	52.8	10.1	17.1	30.2	49.0
LLaMA	7B	35.6	27.3	11.0	2.9	12.8	17.7	33.5
	13B	47.7	31.8	20.3	4.2	15.8	22.0	37.9
	33B	58.7	37.5	42.3	7.1	21.7	30.2	50.0
	65B	63.7	40.4	54.4	10.6	23.7	37.7	58.4
LLAMA 2	7B	46.8	32.5	16.7	3.3	12.8	20.8	38.2
	13B	55.0	41.4	29.6	5.0	18.9	30.3	45.6
	34B	62.6	-	42.2	6.2	22.6	33.0	44.1
	70B	69.8	50.1	63.3	13.5	29.9	45.0	64.9
StableBeluga2	70B	68.6	51.4	69.6	14.6	28.0	11.4	69.3
QWEN	1.8B	44.6	54.7	21.2	5.6	17.1	14.8	28.2
	7B	58.2	63.5	51.7	11.6	29.9	31.6	45.0
	14B	**66.3**	**72.1**	**61.3**	**24.8**	**32.3**	**40.8**	**53.4**

2.4 TRAINING

To train QWEN, we follow the standard approach of autoregressive language modeling, as described in Radford et al. (2018). This involves training the model to predict the next token based on the context provided by the previous tokens. We train models with context lengths of 2048. To create batches of data, we shuffle and merge the documents, and then truncate them to the specified context lengths. To improve computational efficiency and reduce memory usage, we employ Flash Attention in the attention modules (Dao et al., 2022). We adopt the standard optimizer AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017) for pretraining optimization. We set the hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$. We use a cosine learning rate schedule with a specified peak learning rate for each model size. The learning rate is decayed to a minimum learning rate of 10% of the peak learning rate. All the models are trained with BFloat16 mixed precision for training stability.

2.5 EXPERIMENTAL RESULTS

To evaluate the zero-shot and few-shot learning capabilities of our models, we conduct a thorough benchmark assessment using a series of datasets. We compare QWEN with the most recent open-source base models, including LLaMA (Touvron et al., 2023a), LLaMA2 (Touvron et al., 2023b), MPT (Mosaic ML, 2023), Falcon (Almazrouei et al., 2023), Baichuan2 (Yang et al., 2023), ChatGLM2 (ChatGLM2 Team, 2023), InternLM (InternLM Team, 2023), XVERSE (Inc., 2023b), and StableBeluga2 (Stability AI, 2023). Our evaluation covers a total of 7 popular benchmarks, which are MMLU (5-shot) (Hendrycks et al., 2020), C-Eval (5-shot) (Huang et al., 2023), GSM8K (8-shot) (Cobbe et al., 2021), MATH (4-shot) (Hendrycks et al., 2021), HumanEval (0-shot) (Chen et al., 2021b), MBPP (0-shot) (Austin et al., 2021), and BBH (Big Bench Hard) (3 shot) (Suzgun et al., 2022). We aim to provide a comprehensive summary of the overall performance of our models across these benchmarks.

Table 3: **Results of QWEN on long-context inference using various techniques.** Our experimental findings reveal that the application of our crucial techniques enables the model to consistently achieve low perplexity as the context length increases. This suggests that these techniques play a significant role in enhancing the model's ability to comprehend and generate lengthy texts.

Model	Sequence Length				
	1024	2048	4096	8192	16384
QWEN-7B	4.23	3.78	39.35	469.81	2645.09
+ dynamic_ntk	4.23	3.78	3.59	3.66	5.71
+ dynamic_ntk + logn	4.23	3.78	3.58	3.56	4.62
+ dynamic_ntk + logn + window_attn	4.23	3.78	3.58	3.49	4.32
QWEN-14B	-	3.46	22.79	334.65	3168.35
+ dynamic_ntk + logn + window_attn	-	3.46	3.29	3.18	3.42

In this evaluation, we focus on the base language models without alignment and collect the baselines' best scores from their official results and OpenCompass (OpenCompass Team, 2023). The results are presented in Table 2.

Our experimental results demonstrate that the three QWEN models exhibit exceptional performance across all downstream tasks. It is worth noting that even the larger models, such as LLaMA2-70B, are outperformed by QWEN-14B in 3 tasks. QWEN-7B also performs admirably, surpassing LLaMA2-13B and achieving comparable results to Baichuan2-13B. Notably, despite having a relatively small number of parameters, QWEN-1.8B is capable of competitive performance on certain tasks and even outperforms larger models in some instances. The findings highlight the impressive capabilities of the QWEN models, particularly QWEN-14B, and suggest that smaller models, such as QWEN-1.8B, can still achieve strong performance in certain applications.

To evaluate the effectiveness of context length extension, Table 3 presents the test results on arXiv[3] in terms of perplexity (PPL). These results demonstrate that by combining NTK-aware interpolation, LogN-Scaling, and layer-wise window assignment, we can effectively maintain the performance of our models in the context of over 8192 tokens.

3 ALIGNMENT

Pretrained large language models have been found to be out of sync with human behavior, making them unsuitable for serving as AI assistants in most cases. Recent research has shown that the use of alignment techniques, such as supervised finetuning (SFT) and reinforcement learning from human feedback (RLHF), can significantly improve the ability of language models to engage in natural conversation. In this section, we will delve into the details of how Qwen models have been trained using SFT and RLHF, and evaluate their performance in the context of chat-based assistance.

3.1 SUPERVISED FINETUNING

To gain an understanding of human behavior, the initial step is to carry out supervised finetuning. This process fine-tunes a pre-trained model on chat-style data, which includes both human queries and AI responses. Supervised finetuning is similar to text-to-text transfer, but it is capable of creating a helpful AI assistant due to the intricate and varied nature of the datasets used for finetuning. In the following sections, we will delve into the details of data construction and training methods.

3.1.1 DATA

To enhance the capabilities of our supervised finetuning datasets, we have annotated conversations in multiple styles. While conventional FLAN datasets (Wei et al., 2022a) contain a vast amount of data prompted with questions, instructions, and answers in natural language, our approach takes it a step further by annotating human-style conversations. This practice, inspired by Ouyang et al.

[3]The dataset contains academic papers from https://arxiv.org/

(2022), is aimed at improving the model's helpfulness by focusing on natural language generation for diverse tasks. To ensure the model's ability to generalize to a wide range of scenarios, we specifically excluded data formatted in prompt templates that could potentially limit its capabilities. Furthermore, we have prioritized the safety of the language model by annotating data related to safety concerns such as violence, bias, and pornography. This will enable the model to detect and reject malicious prompts or provide safe answers in such situations.

In addition to data quality, we have observed that the training method can significantly impact the final performance of the model. To achieve this, we utilized the ChatML-style format (OpenAI, 2022), which is a versatile meta language capable of describing both the metadata (such as roles) and the content of a turn. This format enables the model to effectively distinguish between various types of information, including system setup, user inputs, and assistant outputs, among others. By leveraging this approach, we can enhance the model's ability to accurately process and analyze complex conversational data.

3.1.2 TRAINING

Consistent with pretraining, we also apply next-token prediction as the training task for SFT. We apply the loss masks for the system and user inputs. More details are demonstrated in Appendix A.1.1.

The model's training process utilizes the AdamW optimizer, with the following hyperparameters: β_1 set to 0.9, β_2 set to 0.95, and ϵ set to 10^{-8}. The sequence length is limited to 2048, and the batch size is 128. The model undergoes a total of 4000 steps, with the learning rate gradually increased over the first 1430 steps, reaching a peak of 2×10^{-6}. To prevent overfitting, weight decay is applied with a value of 0.1, dropout regularization is set to 0.1, and gradient clipping is enforced with a limit of 1.0.

3.2 REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

While SFT has proven to be effective, we acknowledge that its generalization and creativity capabilities may be limited, and it is prone to overfitting. To address this issue, we have implemented Reinforcement Learning from Human Feedback (RLHF) to further align SFT models with human preferences, following the approaches of Ouyang et al. (2022); Christiano et al. (2017). This process involves training a reward model and using Proximal Policy Optimization (PPO) (Schulman et al., 2017) to conduct policy training.

3.2.1 REWARD MODEL

To create a successful reward model, like building a large language model (LLM), it is crucial to first undergo pretraining and then finetuning. This pretraining process, also known as preference model pretraining (PMP) (Bai et al., 2022b), necessitates a vast dataset of comparison data. This dataset consists of sample pairs, each containing two distinct responses for a single query and their corresponding preferences. Similarly, finetuning is also conducted on this type of comparison data, but with a higher quality due to the presence of quality annotations.

During the fine-tuning phase, we gather a variety of prompts and adjust the reward model based on human feedback for responses from the QWEN models. To ensure the diversity and complexity of user prompts are properly taken into account, we have created a classification system with around 6600 detailed tags and implemented a balanced sampling algorithm that considers both diversity and complexity when selecting prompts for annotation by the reward model (Lu et al., 2023). To generate a wide range of responses, we have utilized QWEN models of different sizes and sampling strategies, as diverse responses can help reduce annotation difficulties and enhance the performance of the reward model. These responses are then evaluated by annotators following a standard annotation guideline, and comparison pairs are formed based on their scores.

In creating the reward model, we utilize the same-sized pre-trained language model QWEN and initiate the PMP process. Subsequently, we fine-tune the PMP model to enhance its performance. It is important to mention that we have incorporated a pooling layer into the original QWEN model to extract the reward for a sentence based on a specific end token. The learning rate for this process has been set to a constant value of 3×10^{-6}, and the batch size is 64. Additionally, the sequence length is set to 2048, and the training process lasts for a single epoch.

Table 4: Test Accuracy of QWEN PMP and reward model on diverse human preference benchmark datasets.

Model	QWEN Helpful-base	QWEN Helpful-online	Anthropic Helpful-base	Anthropic Helpful-online	OpenAI Summ.	Stanford SHP	OpenAI PRM800K
PMP	62.68	61.62	76.52	65.43	69.60	60.05	70.59
RM	74.78	69.71	73.98	64.57	69.99	60.10	70.52

3.2.2 REINFORCEMENT LEARNING

Our Proximal Policy Optimization (PPO) process involves four models: the policy model, value model, reference model, and reward model. Before starting the PPO procedure, we pause the policy model's updates and focus solely on updating the value model for 50 steps. This approach ensures that the value model can adapt to different reward models effectively.

During the PPO operation, we use a strategy of sampling two responses for each query simultaneously. This strategy has proven to be more effective based on our internal benchmarking evaluations. We set the KL divergence coefficient to 0.04 and normalize the reward based on the running mean.

The policy and value models have learning rates of 1×10^{-6} and 5×10^{-6}, respectively. To enhance training stability, we utilize value loss clipping with a clip value of 0.15. For inference, the policy top-p is set to 0.9. Our findings indicate that although the entropy is slightly lower than when top-p is set to 1.0, there is a faster increase in reward, ultimately resulting in consistently higher evaluation rewards under similar conditions.

Additionally, we have implemented a pre-trained gradient to mitigate the alignment tax. Empirical findings indicate that, with this specific reward model, the KL penalty is adequately robust to counteract the alignment tax in benchmarks that are not strictly code or math in nature, such as those that test common sense knowledge and reading comprehension. It is imperative to utilize a significantly larger volume of the pretrained data in comparison to the PPO data to ensure the effectiveness of the pretrained gradient. Additionally, our empirical study suggests that an overly large value for this coefficient can considerably impede the alignment to the reward model, eventually compromising the ultimate alignment, while an overly small value would only have a marginal effect on alignment tax reduction.

3.3 AUTOMATIC AND HUMAN EVALUATION OF ALIGNED MODELS

To showcase the effectiveness of our aligned models, we conduct a comparison with other aligned models on well-established benchmarks, including MMLU (Hendrycks et al., 2020), C-Eval (Huang et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021b), and BBH (Suzgun et al., 2022). Besides the widely used few-shot setting, we test our aligned models in the zero-shot setting to demonstrate how well the models follow instructions. The prompt in a zero-shot setting consists of an instruction and a question without any previous examples in the context. The results of the baselines are collected from their official reports and OpenCompass (OpenCompass Team, 2023).

The results in Table 5 demonstrate the effectiveness of our aligned models in understanding human instructions and generating appropriate responses. QWEN-14B-Chat outperforms all other models except ChatGPT (OpenAI, 2022) and LLAMA 2-CHAT-70B (Touvron et al., 2023b) in all datasets, including MMLU (Hendrycks et al., 2020), C-Eval (Huang et al., 2023), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021b), and BBH (Suzgun et al., 2022). In particular, QWEN's performance in HumanEval, which measures the quality of generated codes, is significantly higher than that of other open-source models.

Moreover, QWEN's performance is consistently better than that of open-source models of similar size, such as LLaMA2 (Touvron et al., 2023b), ChatGLM2 (ChatGLM2 Team, 2023), InternLM (InternLM Team, 2023), and Baichuan2 (Yang et al., 2023). This suggests that our alignment approach, which involves fine-tuning the model on a large dataset of human conversations, has been effective in improving the model's ability to understand and generate human-like language.

Table 5: **Performance of aligned models on widely-used benchmarks.** We report both zero-shot and few-shot performance of the models.

Model	Params	MMLU 0-shot / 5-shot	C-Eval 0-shot / 5-shot	GSM8K 0-shot / 8-shot	HumanEval 0-shot	BBH 0-shot / 3-shot
Proprietary models						
GPT-3.5	-	- / 69.1	- / 52.5	- / 78.2	73.2	- / 70.1
GPT-4	-	- / **83.0**	- / **69.9**	- / **91.4**	**86.6**	- / **86.7**
Open-source models						
ChatGLM2	6B	45.5 / 46.0	50.1 / 52.6	- / 28.8	11.0	- / 32.7
InternLM-Chat	7B	- / 51.1	- / 53.6	- / 33.0	14.6	- / 32.5
Baichuan2-Chat	7B	- / 52.9	- / 55.6	- / 32.8	13.4	- / 35.8
	13B	- / 57.3	- / 56.7	- / 55.3	17.7	- / 49.9
LLAMA 2-CHAT	7B	- / 46.2	- / 31.9	- / 26.3	12.2	- / 35.6
	13B	- / 54.6	- / 36.2	- / 37.1	18.9	- / 40.1
	70B	- / 63.8	- / 44.3	- / 59.3	32.3	- / 60.8
QWEN-CHAT	1.8B	42.4 / 43.9	50.7 / 50.3	27.8 / 19.5	14.6	27.1 / 25.0
	7B	55.8 / 57.0	59.7 / 59.3	50.3 / 54.1	37.2	39.6 / 46.7
	14B	64.6 / **66.5**	69.8 / **71.7**	**60.1** / 59.3	**43.9**	46.9 / **58.7**

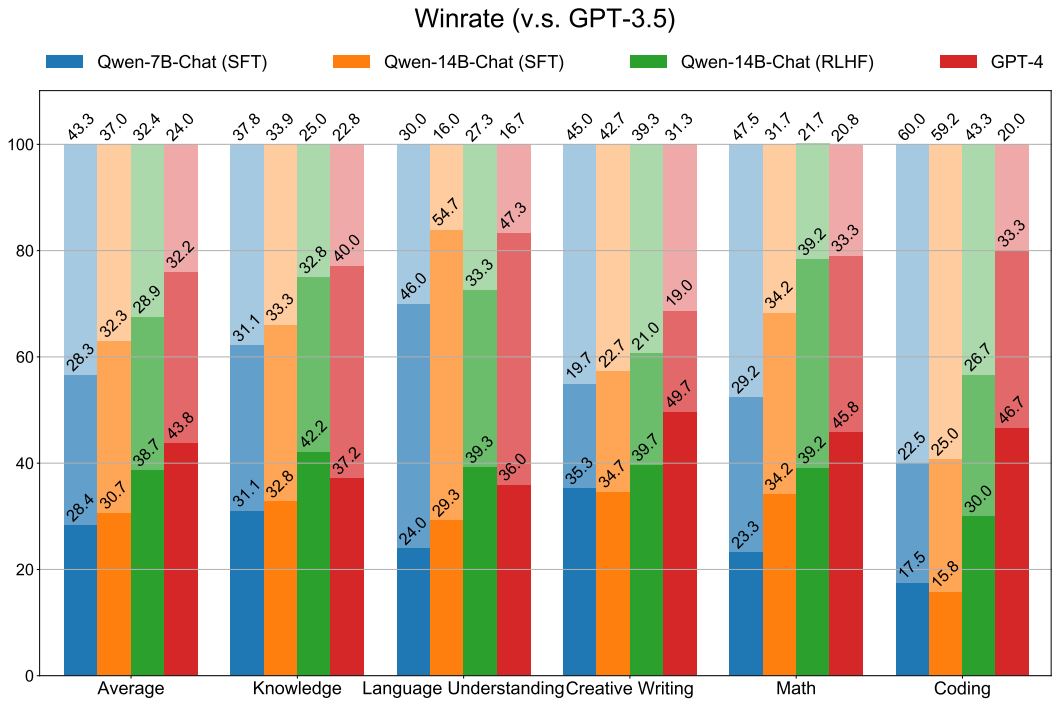

Figure 4: **Results of the human evaluation for chat models.** We compare Qwen-7B (SFT), Qwen-14B (SFT), Qwen-14B (RLHF), as well as GPT-4 against GPT-3.5. Each bar segment represents the percentage of wins, ties, and losses, from bottom to top. On average, the RLHF model outperforms the SFT model and falls behind GPT-4 by a relatively small margin.

Despite this, we have reservations about the ability of traditional benchmark evaluation to accurately measure the performance and potential of chat models trained with alignment techniques in today's landscape. The results mentioned earlier provide some evidence of our competitive standing, but we believe that it is crucial to develop new evaluation methods specifically tailored to aligned models.

We believe that human evaluation is crucial, which is why we have created a carefully curated dataset for this purpose. Our process involved collecting 300 instructions in Chinese that covered a wide range of topics, including knowledge, language understanding, creative writing, coding, and mathematics. To evaluate the performance of different models, we chose the SFT version of QWEN-CHAT-7B and the SFT and RLHF version of QWEN-CHAT-14B, and added two strong baselines, GPT-3.5 and GPT-4[4], for comparison. For each instruction, we asked three annotators to rank the model responses by the overall score of helpfulness, informativeness, validity, and other relevant factors. Our dataset and evaluation methodology provides a comprehensive and rigorous assessment of the capabilities of different language models in various domains.

Figure 4 illustrates the win rates of the various models. For each model, we report the percentage of wins, ties, and losses against GPT-3.5, with the segments of each bar from bottom to top representing these statistics. The experimental results clearly demonstrate that the RLHF model outperforms the SFT models by significant margins, indicating that RLHF can encourage the model to generate responses that are more preferred by humans. In terms of overall performance, we find that the RLHF model significantly outperforms the SFT models, slightly falling behind GPT-4. This indicates the effectiveness of RLHF for aligning to human preference. To provide a more comprehensive understanding of the models' performance, we include a case study with examples from different models in Appendix A.2.2.

3.4 TOOL USE, CODE INTERPRETER, AND AGENT

Table 6: Performance of QWEN on the in-house Chinese benchmark that evaluates its ability to use unseen tools via ReAct prompting.

Model	Params	Tool Selection (Acc.↑)	Tool Input (Rouge-L↑)	False Positive Error (%)↓
GPT-4	-	95	90	15.0
GPT-3.5	-	85	88	75.0
QWEN-CHAT	1.8B	92	89	19.3
	7B	**98**	91	7.3
	14B	**98**	**93**	**2.4**

The QWEN models, which are designed to be versatile, have the remarkable ability to assist with (semi-)automating daily tasks by leveraging their skills in tool-use and planning. As such, they can serve as agents or copilots to help streamline various tasks. We explore QWEN's proficiency in the following areas:

- Utilizing unseen tools through ReAct prompting (Yao et al., 2022) (see Table 6).
- Using a Python code interpreter to enhance math reasoning, data analysis, and more (see Table 7 and Table 8).
- Functioning as an agent that accesses Hugging Face's extensive collection of multimodal models while engaging with humans (see Table 9).

To enhance QWEN's capabilities as an agent or copilot, we employ the self-instruct (Wang et al., 2023c) strategy for supervised fine-tuning (SFT). Specifically, we utilize the in-context learning capability of QWEN for self-instruction. By providing a few examples, we can prompt QWEN to generate more relevant queries and generate outputs that follow a specific format, such as ReAct. We then apply rules and involve human annotators to filter out any noisy samples. Afterwards, the samples are incorporated into QWEN's training data, resulting in an updated version of QWEN that is more dependable for self-instruction. We iterate through this process multiple times until we gather

[4]To obtain the results from the models, we use the OpenAI APIs of GPT-3.5-turbo-0613 and GPT-4-0613.

Table 7: The proportion of code generated by QWEN that is executable on the in-house evaluation benchmark for Code Interpreter. This benchmark examines QWEN's coding proficiency in math problem solving, data visualization, and general purposes. CODE LLAMA underperforms on visualization tasks because it hallucinates non-existent columns solely based on CSV file names (see Figure 5).

Model	Params	Category			
		Math (%)	Visualization (%)	General (%)	All (%)
GPT-4	-	91.9	85.9	82.8	86.8
GPT-3.5	-	89.2	65.0	74.1	72.9
LLAMA 2-CHAT	7B	41.9	33.1	24.1	33.6
	13B	50.0	40.5	48.3	44.4
CODE LLAMA-INSTRUCT	7B	85.1	54.0	70.7	65.1
	13B	93.2	55.8	74.1	68.8
InternLM-Chat	7B v1.1	78.4	44.2	62.1	56.3
	20B	70.3	44.2	65.5	54.9
QWEN-CHAT	1.8B	33.8	30.1	8.6	26.8
	7B	82.4	64.4	67.2	70.2
	14B	89.2	84.1	65.5	81.7

Table 8: Correctness of the final response on the in-house evaluation benchmark for Code Interpreter. Visualization-Hard tasks involve planning multiple steps, while Visualization-Easy tasks do not. Visualization-All measures both types of tasks. CODE LLAMA excels in performing Visualization-Easy tasks but tends to underperform in Visualization-Hard tasks, due to its inclination to hallucinate non-existent columns based on the name of a CSV file (see Figure 5).

Model	Params	Category			
		Math (%)	Vis.-Hard (%)	Vis.-Easy (%)	Vis.-All (%)
GPT-4	-	82.8	66.7	60.8	63.8
GPT-3.5	-	47.3	33.3	55.7	44.2
LLAMA 2-CHAT	7B	3.9	14.3	39.2	26.4
	13B	8.3	8.3	40.5	23.9
CODE LLAMA-INSTRUCT	7B	14.3	26.2	60.8	42.9
	13B	28.2	27.4	62.0	44.2
InternLM-Chat	7B v1.1	28.5	4.8	40.5	22.1
	20B	34.6	21.4	45.6	33.1
QWEN-CHAT	1.8B	14.7	3.6	20.3	11.7
	7B	41.9	40.5	54.4	47.2
	14B	58.4	53.6	59.5	56.4

Table 9: Results of QWEN-Chat on the Hugging Face Agent benchmark.

Task	Model	Params	Metric		
			Tool Selection ↑	Tool Used ↑	Code Correctness ↑
Run Mode	GPT-4	-	100	100	97.4
	GPT-3.5	-	95.4	96.3	87.0
	Starcoder-Base	15B	86.1	87.0	68.9
	Starcoder	15B	87.0	88.0	68.9
	QWEN-CHAT	1.8B	85.2	84.3	61.1
		7B	87.0	87.0	71.5
		14B	93.5	94.4	87.0
Chat Mode	GPT-4	-	97.9	97.9	98.5
	GPT-3.5	-	97.3	96.8	89.6
	Starcoder-Base	15B	97.9	97.9	91.1
	Starcoder	15B	97.9	97.9	89.6
	QWEN-CHAT	1.8B	93.6	93.6	73.2
		7B	94.7	94.7	85.1
		14B	97.9	97.9	95.5

an ample number of samples that possess both exceptional quality and a wide range of diversity. As a result, our final collection consists of around 2000 high-quality samples.

During the fine-tuning process, we mix these high-quality samples with all the other general-purpose SFT samples, rather than introducing an additional training stage. By doing so, we are able to retain essential general-purpose capabilities that are also pertinent for constructing agent applications.

Using Tools via ReAct Prompting We have created and made publicly available a benchmark for evaluating QWEN's ability to call plugins, tools, functions, or APIs using ReAct Prompting (see Qwen Team, Alibaba Cloud, 2023b). To ensure fair evaluation, we have excluded any plugins that were included in QWEN's training set from the evaluation set. The benchmark assesses the model's accuracy in selecting the correct plugin from a pool of up to five candidates, as well as the plausibility of the parameters passed into the plugin and the frequency of false positives. In this evaluation, a false positive occurs when the model incorrectly invokes a plugin in response to a query, despite not being required to do so.

The results presented in Table 6 demonstrate that QWEN consistently achieves higher accuracy in identifying the relevance of a query to the available tools as the model size increases. However, the table also highlights that beyond a certain point, there is little improvement in performance when it comes to selecting the appropriate tool and providing relevant arguments. This suggests that the current preliminary benchmark may be relatively easy and may require further enhancement in future iterations. It is worth noting that GPT-3.5 stands out as an exception, displaying suboptimal performance on this particular benchmark. This could potentially be attributed to the fact that the benchmark primarily focuses on the Chinese language, which may not align well with GPT-3.5's capabilities. Additionally, we observe that GPT-3.5 tends to attempt to use at least one tool, even if the query cannot be effectively addressed by the provided tools.

Using Code Interpreter for Math Reasoning and Data Analysis The Python code interpreter is widely regarded as a powerful tool for augmenting the capabilities of an LLM agent. It is worth investigating whether QWEN can harness the full potential of this interpreter to enhance its performance in diverse domains, such as mathematical reasoning and data analysis. To facilitate this exploration, we have developed and made publicly available a benchmark that is specifically tailored for this purpose (see Qwen Team, Alibaba Cloud, 2023a).

The benchmark encompasses three primary categories of tasks: math problem-solving, data visualization, and other general-purpose tasks like file post-processing and web crawling. Within the

visualization tasks, we differentiate between two levels of difficulty. The easier level can be achieved by simply writing and executing a single code snippet without the need for advanced planning skills. However, the more challenging level requires strategic planning and executing multiple code snippets in a sequential manner. This is because the subsequent code must be written based on the output of the previous code. For example, an agent may need to examine the structure of a CSV file using one code snippet before proceeding to write and execute additional code to create a plot.

Regarding evaluation metrics, we consider both the executability and correctness of the generated code. To elaborate on the correctness metrics, for math problems, we measure accuracy by verifying if the ground truth numerical answer is present in both the code execution result and the final response. When it comes to data visualization, we assess accuracy by utilizing QWEN-VL (Bai et al., 2023), a powerful multimodal language model. QWEN-VL is capable of answering text questions paired with images, and we rely on it to confirm whether the image generated by the code fulfills the user's request.

The results regarding executability and correctness are presented in Table 7 and Table 8, respectively. It is evident that CODE LLAMA generally outperforms LLAMA 2, its generalist counterpart, which is not surprising since this benchmark specifically requires coding skills. However, it is worth noting that specialist models that are optimized for code synthesis do not necessarily outperform generalist models. This is due to the fact that this benchmark encompasses various skills beyond coding, such as abstracting math problems into equations, understanding language-specified constraints, and responding in the specified format such as ReAct. Notably, QWEN-7B-CHAT and QWEN-14B-CHAT surpass all other open-source alternatives of similar scale significantly, despite being generalist models.

Serving as a Hugging Face Agent Hugging Face provides a framework called the Hugging Face Agent or Transformers Agent (Hugging Face, 2023), which empowers LLM agents with a curated set of multimodal tools, including speech recognition and image synthesis. This framework allows an LLM agent to interact with humans, interpret natural language commands, and employ the provided tools as needed.

To evaluate QWEN's effectiveness as a Hugging Face agent, we utilized the evaluation benchmarks offered by Hugging Face. The results are presented in Table 9. The evaluation results reveal that QWEN performs quite well in comparison to other open-source alternatives, only slightly behind the proprietary GPT-4, demonstrating QWEN's competitive capabilities.

4 CODE-QWEN: SPECIALIZED MODEL FOR CODING

Training on domain-specific data has been shown to be highly effective, particularly in the case of code pretraining and finetuning. A language model that has been reinforced with training on code data can serve as a valuable tool for coding, debugging, and interpretation, among other tasks. In this work, we have developed a series of generalist models using pretraining and alignment techniques. Building on this foundation, we have created domain-specific models for coding by leveraging the base language models of QWEN. We have implemented continued pretraining on code data for CODE-QWEN and then applied supervised finetuning to create CODE-QWEN-CHAT. The code models, CODE-QWEN-14B and CODE-QWEN-7B, are based on base language models with 14 billion and 7 billion parameters.

4.1 CODE PRETRAINING

Unlike previous approaches that focused solely on pretraining on code data (Li et al., 2022; 2023d), we take a different approach (Rozière et al., 2023) by starting with our base language models, CODE-QWEN-14B-CHAT and CODE-QWEN-7B-CHAT, and then continuing to pretrain on a combination of text and code data. We believe that relying solely on code data for pretraining can result in a significant loss of the ability to function as a versatile assistant. Additionally, incorporating data from a diverse range of domains can help enhance the models' ability to understand and generate code, as well as bridge the gap between language and coding. We continue to pretrain the models on a total of around 90 billion tokens.

In the pretraining process, we initialize the model by corresponding base language models, CODE-QWEN-14B-CHAT and CODE-QWEN-7B-CHAT. Most applications depend on specialized models for coding may lead to long contextual scenarios, such as tool use and code interpreter in Section 3.4. In that case, we train models with context lengths of 8192. Similar to base model training in Section 2.4, we employ Flash Attention (Dao et al., 2022) in the attention modules, and adopt the standard optimizer AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017), setting $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$. We set the learning rate as 3.0×10^{-5} for CODE-QWEN-7B and 6.0×10^{-5} CODE-QWEN-14B, with 3% warm up iterations and no learning rate decays.

4.2 CODE SUPERVISED FINE-TUNING

After conducting a series of empirical experiments, we have determined that the multi-stage SFT strategy yields the best performance compared to other methods. In the supervised fine-tuning stage, the models CODE-QWEN-7B-CHAT and CODE-QWEN-14B-CHAT initialized by the code foundation model CODE-QWEN-7B-CHAT and CODE-QWEN-14B-CHAT are optimized by the AdamW (Kingma & Ba, 2014; Loshchilov & Hutter, 2017) optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$) with a learning rate of 1.0×10^{-5} and 2.0×10^{-6}, respectively. The learning rate increases to the peaking value with the cosine learning rate schedule (3% warm-up steps) and remains constant.

4.3 EVALUATION

Our CODE-QWEN models have been compared with both proprietary and open-source language models, as shown in Tables 11 and 10. These tables present the results of our evaluation on the test sets of Humaneval (Chen et al., 2021b), MBPP (Austin et al., 2021), and the multi-lingual code generation benchmark HUMANEVALPACK (Muennighoff et al., 2023). The comparison is based on the pass@1 performance of the models on these benchmark datasets. The results of this comparison are clearly demonstrated in Tables 10 and 11.

Our analysis reveals that specialized models, specifically CODE-QWEN and CODE-QWEN-CHAT, significantly outperform previous baselines with similar parameter counts, such as OCTOGEEX (Muennighoff et al., 2023), InstructCodeT5+ (Wang et al., 2023d), and CodeGeeX2 (Zheng et al., 2023). In fact, these models even rival the performance of larger models like Starcoder (Li et al., 2023d) and WizardCoder-15B (Luo et al., 2023b).

When compared to some of the extremely large-scale closed-source models, CODE-QWEN and CODE-QWEN-CHAT demonstrate clear advantages in terms of pass@1. However, it is important to note that these models fall behind the state-of-the-art methods, such as GPT-4, in general. Nonetheless, with the continued scaling of both model size and data size, we believe that this gap can be narrowed in the near future.

It is crucial to emphasize that the evaluations mentioned previously are insufficient for grasping the full extent of the strengths and weaknesses of the models. In our opinion, it is crucial to develop more rigorous tests to enable us to accurately assess our relative performance in comparison to GPT-4.

5 MATH-QWEN: SPECIALIZED MODEL FOR MATHEMATICS REASONING

We have created a mathematics-specialized model series called MATH-QWEN-CHAT, which is built on top of the QWEN pretrained language models. Specifically, we have developed assistant models that are specifically designed to excel in arithmetic and mathematics and are aligned with human behavior. These models are referred to as MATH-QWEN-CHAT. We are releasing two versions of this model series, MATH-QWEN-14B-CHAT and MATH-QWEN-7B-CHAT, which have 14 billion and 7 billion parameters, respectively.

5.1 TRAINING

We carry out math SFT on our augmented math instructional dataset for mathematics reasoning, and therefore we obtain the chat model, MATH-QWEN-CHAT, directly. Owing to shorter average lengths of the math SFT data, we use a sequence length of 1024 for faster training. Most user inputs

Table 10: Results of pass@1 (%) on HumanEval and MBPP. Most scores are retrieved from the papers of StarCoder (Li et al., 2023d), CodeT5+ (Wang et al., 2023d), WizardCoder (Luo et al., 2023b) and CODE LLAMA (Rozière et al., 2023).

Model	Params	HumanEval	MBPP
Proprietary models			
PaLM	540B	26.2	36.8
PaLM-Coder	540B	36.0	47.0
PaLM 2-S	-	37.6	50.0
Code-Cushman-001	-	33.5	45.9
Code-Davinci-002	-	47.0	58.1
GPT-3.5	-	73.2	-
GPT-4	-	86.6	-
Open-source models			
LLAMA 2	7B	12.2	20.8
	13B	20.1	27.6
	34B	22.6	33.8
	70B	30.5	45.4
CodeGen-Multi	16B	18.3	20.9
CodeGen-Mono	16B	29.3	35.3
CodeGeeX2	6B	35.9	-
StarCoder-Prompted	15B	40.8	49.5
CodeT5+	16B	30.9	-
InstructCodeT5+	16B	35.0	-
CODE LLAMA	7B	33.5	41.4
	13B	36.0	47.0
	34B	48.8	55.0
CODE LLAMA-INSTRUCT	7B	34.8	44.4
	13B	42.7	49.4
	34B	41.5	57.0
CODE LLAMA-PYTHON	7B	38.4	47.6
	13B	43.3	49.0
	34B	53.7	56.2
UNNATURAL CODE LLAMA	34B	62.2	61.2
WizardCoder-Python	13B	64.0	**55.6**
	34B	73.2	61.2
QWEN-CHAT	7B	37.2	35.8
	14B	43.9	46.4
CODE-QWEN	7B	40.2	41.8
	14B	45.1	51.4
CODE-QWEN-CHAT	7B	43.3	44.2
	14B	**66.4**	52.4

Table 11: **Zero-shot pass@1 (%) performance on the HUMANEVALPACK (synthesize) benchmark.** The baseline results are partly from OCTOPACK (Muennighoff et al., 2023)

Model	Params	Programming Language						
		Python	JavaScript	Java	Go	C++	Rust	Avg.
Proprietary models								
GPT-4	-	86.6	82.9	81.7	72.6	78.7	67.1	78.3
Open-source models								
InstructCodeT5+	16B	37.0	18.9	17.4	9.5	19.8	0.3	17.1
StarChat-β	15B	33.5	31.4	26.7	25.5	26.6	14.0	26.3
StarCoder	15B	33.6	30.8	30.2	17.6	31.6	21.8	27.6
CodeGeeX2	6B	35.9	32.2	30.8	22.5	29.3	18.1	28.1
OCTOGEEX	6B	44.7	33.8	36.9	21.9	32.3	15.7	30.9
OCTOCODER	15B	46.2	39.2	38.2	30.4	35.6	23.4	35.5
WizardCoder	15B	59.8	49.5	36.1	36.4	40.9	20.2	40.5
QWEN-CHAT	7B	37.2	23.2	32.9	20.7	22.0	9.1	24.2
	14B	43.9	38.4	42.7	34.1	24.4	18.9	33.7
CODE-QWEN	7B	40.2	40.4	40.2	26.2	20.7	15.8	30.6
	14B	45.1	51.8	57.3	39.6	18.2	20.7	38.8
CODE-QWEN-CHAT	7B	43.3	41.5	49.4	29.3	32.9	20.1	36.1
	14B	**66.4**	**58.5**	**56.1**	**47.6**	**54.2**	**28.7**	**51.9**

Table 12: **Results of models on mathematical reasoning**. We report the accuracy of QWEN for all benchmarks using greedy decoding. For MATH, we are reporting QWEN's performances on the test set from Lightman et al. (2023).

Model	Params	GSM8K	MATH	Math401	Math23K
Proprietary models					
GPT-4	-	**92.0**	**42.5**	83.5	74.0
GPT-3.5	-	80.8	34.1	75.1	60.0
Minerva	8B	16.2	14.1	-	-
	62B	52.4	27.6	-	-
	540B	58.8	33.6	-	-
Open-source models					
LLaMA-1 RFT	7B	46.5	5.2	-	-
	13B	52.1	5.1	-	-
WizardMath	7B	54.9	10.7	-	-
	13B	63.9	14.0	-	-
	70B	81.6	22.7	-	-
GAIRMath-Abel	7B	59.7	13.0	-	-
	13B	66.4	17.3	-	-
	70B	83.6	28.3	-	-
QWEN-CHAT	7B	50.3	6.8	57.4	51.2
	14B	60.1	18.4	70.1	67.0
MATH-QWEN-CHAT	7B	62.5	17.2	80.8	75.4
	14B	69.8	24.2	**85.0**	**78.4**

in the math SFT dataset are examination questions, and it is easy for the model to predict the input format and it is meaningless for the model to predict the input condition and numbers which could be random. Thus, we mask the inputs of the system and user to avoid loss computation on them and find masking them accelerates the convergence during our preliminary experiments. For optimization, we use the AdamW optimizer with the same hyperparameters of SFT except that we use a peak learning rate of 2×10^{-5} and a training step of $50\,000$.

5.2 Evaluation

We evaluate models on the test sets of GSM8K (Grade school math) (Cobbe et al., 2021), MATH (Challenging competition math problems) (Hendrycks et al., 2021), Math401 (Arithmetic ability) (Yuan et al., 2023b), and Math23K (Chinese grade school math) (Wang et al., 2017). We compare MATH-QWEN with proprietary models ChatGPT and Minerva (Lewkowycz et al., 2022) and open-sourced math-specialized model RFT (Yuan et al., 2023a), WizardMath (Luo et al., 2023a), and GAIRMath-Abel (Chern et al., 2023a) in Table 12. MATH-QWEN-CHAT models show better math reasoning and arithmetic abilities compared to open-sourced models and QWEN-CHAT models of similar sizes. Compared to proprietary models, MATH-QWEN-CHAT-7B outperforms Minerva-8B in MATH. MATH-QWEN-14B-CHAT is chasing Minerva-62B and GPT-3.5 in GSM8K and MATH and delivers better performance on arithmetic ability and Chinese math problems.

6 Related Work

6.1 Large Language Models

The excitement of LLM began with the introduction of the Transformer architecture (Vaswani et al., 2017), which was then applied to pretraining large-scale data by researchers such as Radford et al. (2018); Devlin et al. (2018); Liu et al. (2019). These efforts led to significant success in transfer learning, with model sizes growing from 100 million to over 10 billion parameters (Raffel et al., 2020; Shoeybi et al., 2019).

In 2020, the release of GPT-3, a massive language model that is 10 times larger than T5, demonstrated the incredible potential of few-shot and zero-shot learning through prompt engineering and in-context learning, and later chain-of-thought prompting (Wei et al., 2022c). This success has led to a number of studies exploring the possibilities of further scaling these models (Scao et al., 2022; Zhang et al., 2022; Du et al., 2021; Zeng et al., 2022; Lepikhin et al., 2020; Fedus et al., 2022; Du et al., 2022; Black et al., 2022; Rae et al., 2021; Hoffmann et al., 2022; Chowdhery et al., 2022; Thoppilan et al., 2022). As a result, the community has come to view these large language models as essential foundations for downstream models (Bommasani et al., 2021).

The birth of ChatGPT (OpenAI, 2022) and the subsequent launch of GPT-4 (OpenAI, 2023) marked two historic moments in the field of artificial intelligence, demonstrating that large language models (LLMs) can serve as effective AI assistants capable of communicating with humans. These events have sparked interests among researchers and developers in building language models that are aligned with human values and potentially even capable of achieving artificial general intelligence (AGI) (Anil et al., 2023; Anthropic, 2023a;b).

One notable development in this area is the emergence of open-source LLMs, specifically LLaMA (Touvron et al., 2023a) and LLAMA 2 (Touvron et al., 2023b), which have been recognized as the most powerful open-source language models ever created. This has led to a surge of activity in the open-source community (Wolf et al., 2019), with a series of large language models being developed collaboratively to build upon this progress (Mosaic ML, 2023; Almazrouei et al., 2023; ChatGLM2 Team, 2023; Yang et al., 2023; InternLM Team, 2023).

6.2 Alignment

The community was impressed by the surprising effectiveness of alignment on LLMs. Previously, LLMs without alignment often struggle with issues such as repetitive generation, hallucination, and deviation from human preferences. Since 2021, researchers have been diligently working on developing methods to enhance the performance of LLMs in downstream tasks (Wei et al., 2022a;

Sanh et al., 2021; Longpre et al., 2023; Chung et al., 2022; Muennighoff et al., 2022). Furthermore, researchers have been actively exploring ways to align LLMs with human instructions (Ouyang et al., 2022; Askell et al., 2021; Bai et al., 2022b;c). One major challenge in alignment research is the difficulty of collecting data. While OpenAI has utilized its platform to gather human prompts or instructions, it is not feasible for others to collect such data.

However, there has been some progress in this area, such as the self-instruct approach proposed in Wang et al. (2023c). This innovative work offers a potential solution to the data collection problem in alignment research. As a result, there has been a surge in open-source chat data, including Alpaca (Taori et al., 2023), MOSS (Sun et al., 2023a), Dolly (Conover et al., 2023), Evol-Instruct (Xu et al., 2023b), and others (Sun et al., 2023b; Xu et al., 2023a;c; Chen et al., 2023b; Ding et al., 2023; Ji et al., 2023; Yang, 2023). Similarly, there has been an increase in open-source chat models, such as Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), Guanaco (Dettmers et al., 2023), MOSS (Sun et al., 2023a), WizardLM (Xu et al., 2023b), and others (Xu et al., 2023c; Chen et al., 2023b; Ding et al., 2023; Wang et al., 2023b).

To train an effective chat model, available solutions are mostly based on SFT and RLHF (Ouyang et al., 2022). While SFT is similar to pretraining, it focuses on instruction following using the aforementioned data. However, for many developers, the limited memory capacity is a major obstacle to further research in SFT. As a result, parameter-efficient tuning methods, such as LoRA (Hu et al., 2021) and Q-LoRA (Dettmers et al., 2023), have gained popularity in the community. LoRA tunes only low-rank adapters, while Q-LoRA builds on LoRA and utilizes 4-bit quantized LLMs and paged attention (Dettmers et al., 2022; Frantar et al., 2022; Kwon et al., 2023). In terms of RLHF, recent methods such as PPO (Schulman et al., 2017; Touvron et al., 2023b) have been adopted, but there are also alternative techniques aimed at addressing the complexity of optimization, such as RRHF (Yuan et al., 2023c), DPO (Rafailov et al., 2023), and PRO (Song et al., 2023). Despite the ongoing debate about the effectiveness of RLHF, more evidence is needed to understand how it enhances the intelligence of LLMs and what potential drawbacks it may have.

6.3 Tool Use and Agents

LLM's planning function allows for the invocation of tools, such as APIs or agent capabilities, through in-context learning, as demonstrated by Schick et al. (2023). Yao et al. (2022) introduced ReAct, a generation format that enables the model to generate thoughts on which tool to use, accept input from API observations, and generate a response. GPT-3.5 and GPT-4, when prompted with few shots, have shown consistent and impressive performance. In addition to tool usage, LLMs can utilize external memory sources like knowledge bases (Hu et al., 2023; Zhong et al., 2023b) or search engines (Nakano et al., 2021; Liu et al., 2023b) to generate more accurate and informative answers. This has led to the popularity of frameworks like LangChain (LangChain, Inc., 2023). The research on LLMs for tool use has also sparked interest in building agents with LLM capabilities, such as agents that can call different AI models (Shen et al., 2023; Li et al., 2023a), embodied lifelong learning or multimodal agents (Wang et al., 2023a; Driess et al., 2023), and multiple agents interacting with each other and even building a micro-society (Chen et al., 2023a; Li et al., 2023b; Xu et al., 2023d; Hong et al., 2023).

6.4 LLM for Coding

Previous research has demonstrated that LLMs possess remarkable capabilities in code understanding and generation, particularly those with massive numbers of parameters (Chowdhery et al., 2022; Anil et al., 2023; Rae et al., 2021; Hoffmann et al., 2022). Moreover, several LLMs have been pre-trained, continued pre-trained, or fine-tuned on coding-related data, which has resulted in significantly improved performance compared to general-purpose LLMs. These models include Codex Chen et al. (2021a), AlphaCode (Li et al., 2022), SantaCoder (Allal et al., 2023), Starcoder-Base (Li et al., 2023d), InCoder (Fried et al., 2022), CodeT5 (Wang et al., 2021), CodeGeeX (Zheng et al., 2023), and Code LLaMA (Rozière et al., 2023). In addition to these models, recent studies have focused on developing specialized alignment techniques for coding, such as Code Llama-Instruct (Rozière et al., 2023) and StarCoder (Li et al., 2023d). These models can assist developers in various code-related tasks, including code generation (Chen et al., 2021b; Austin et al., 2021), code completion (Zhang et al., 2023a), code translation (Szafraniec et al., 2023), bug fixing (Muennighoff et al., 2023), code

refinement (Liu et al., 2023c), and code question answering (Liu & Wan, 2021). In a word, LLMs have the potential to revolutionize the field of coding by providing developers with powerful tools for code comprehension, generation, and related tasks.

6.5 LLM FOR MATHEMATICS

LLMs with a certain model scale have been found to possess the ability to perform mathematical reasoning (Wei et al., 2022b; Suzgun et al., 2022). In order to encourage LLMs to achieve better performance on math-related tasks, researchers have employed techniques such as chain-of-thought prompting (Wei et al., 2022c) and scratchpad (Nye et al., 2021), which have shown promising results. Additionally, self-consistency (Wang et al., 2022) and least-to-most prompting (Zhou et al., 2022) have further improved the performance of these models on these tasks. However, prompt engineering is a time-consuming process that requires a lot of trial and error, and it is still difficult for LLMs to consistently perform well or achieve satisfactory results in solving mathematical problems. Moreover, simply scaling the data and model size is not an efficient way to improve a model's mathematical reasoning abilities. Instead, pretraining on math-related corpora has been shown to consistently enhance these capabilities (Hendrycks et al., 2021; Lewkowycz et al., 2022; Taylor et al., 2022; Lightman et al., 2023). Additionally, fine-tuning on math-related instruction-following datasets (Si et al., 2023; Yuan et al., 2023a; Luo et al., 2023a; Yue et al., 2023; Chern et al., 2023a; Yu et al., 2023), has also been effective and more cost-effective than math-specific pretraining. Despite their limitations in terms of accuracy, LLMs still have significant potential to assist users with practical mathematical problems. There is ample scope for further development in this area.

7 CONCLUSION

In this report, we present the QWEN series of large language models, which showcase the latest advancements in natural language processing. With 14B, 7B, and 1.8B parameters, these models have been pre-trained on massive amounts of data, including trillions of tokens, and fine-tuned using cutting-edge techniques such as SFT and RLHF. Additionally, the QWEN series includes specialized models for coding and mathematics, such as CODE-QWEN, CODE-QWEN-CHAT, and MATH-QWEN-CHAT, which have been trained on domain-specific data to excel in their respective fields. Our results demonstrate that the QWEN series is competitive with existing open-source models and even matches the performance of some proprietary models on comprehensive benchmarks and human evaluation.

We believe that the open access of QWEN will foster collaboration and innovation within the community, enabling researchers and developers to build upon our work and push the boundaries of what is possible with language models. By providing these models to the public, we hope to inspire new research and applications that will further advance the field and contribute to our understanding of the variables and techniques introduced in realistic settings. In a nutshell, the QWEN series represents a major milestone in our development of large language models, and we are excited to see how it will be used to drive progress and innovation in the years to come.

REFERENCES

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, Manan Dey, et al. SantaCoder: Don't reach for the stars! *arXiv preprint arXiv:2301.03988*, 2023.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. Falcon-40B: An open large language model with state-of-the-art performance, 2023.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. PaLM 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Anthropic. Introducing Claude, 2023a. URL https://www.anthropic.com/index/introducing-claude.

Anthropic. Claude 2. Technical report, Anthropic, 2023b. URL https://www-files.anthropic.com/production/images/Model-Card-Claude-2.pdf.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

AutoGPT. AutoGPT: The heart of the open-source agent ecosystem, 2023. URL https://github.com/Significant-Gravitas/Auto-GPT.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

Jinze Bai, Rui Men, Hao Yang, Xuancheng Ren, Kai Dang, Yichang Zhang, Xiaohuan Zhou, Peng Wang, Sinan Tan, An Yang andf Zeyu Cui, Yu Han, Shuai Bai, Wenbin Ge, Jianxin Ma, Junyang Lin, Jingren Zhou, and Chang Zhou. OFASys: A multi-modal multi-task learning system for building generalist models. *CoRR*, abs/2212.04408, 2022a. doi: 10.48550/arXiv.2212.04408. URL https://doi.org/10.48550/arXiv.2212.04408.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *CoRR*, abs/2308.12966, 2023. doi: 10.48550/arXiv.2308.12966. URL https://doi.org/10.48550/arXiv.2308.12966.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022b.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022c.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. GPT-NeoX-20B: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*, 2022.

bloc97. NTK-aware scaled RoPE allows LLaMA models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation., 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

ChatGLM2 Team. ChatGLM2-6B: An open bilingual chat LLM, 2023. URL https://github.com/THUDM/ChatGLM2-6B.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison 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, Joshua 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. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021a. URL https://arxiv.org/abs/2107.03374.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, 2023a.

Zhihong Chen, Feng Jiang, Junying Chen, Tiannan Wang, Fei Yu, Guiming Chen, Hongbo Zhang, Juhao Liang, Chen Zhang, Zhiyi Zhang, et al. Phoenix: Democratizing ChatGPT across languages. *arXiv preprint arXiv:2304.10453*, 2023b.

Ethan Chern, Haoyang Zou, Xuefeng Li, Jiewen Hu, Kehua Feng, Junlong Li, and Pengfei Liu. Generative ai for math: Abel. https://github.com/GAIR-NLP/abel, 2023a.

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*, 2023b.

David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7654–7664, 2022.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 2924–2936. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1300. URL https://doi.org/10.18653/v1/n19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.

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

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free Dolly: Introducing the world's first truly open instruction-tuned LLM, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng, Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. *arXiv preprint arXiv:2305.06500*, 2023.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pp. 933–941. PMLR, 2017.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 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. *arXiv preprint arXiv:1810.04805*, 2018.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. GLaM: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. GLM: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1): 5232–5270, 2022.

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

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida I. Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. *ArXiv*, abs/2204.05999, 2022.

Google. An important next step on our AI journey, 2023. URL `https://blog.google/technology/ai/bard-google-ai-search-updates/`.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with Gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL `http://arxiv.org/abs/1606.08415`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*, 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 preprint arXiv:2106.09685*, 2021.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence S. Moss. OCNLI: original chinese natural language inference. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 3512–3526. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.314. URL `https://doi.org/10.18653/v1/2020.findings-emnlp.314`.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, et al. C-Eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023.

Hugging Face. Transformers agents, 2023. URL https://huggingface.co/docs/transformers/transformers_agents.

Baichuan Inc. Baichuan-7B: A large-scale 7B pretraining language model developed by BaiChuan-Inc, 2023a. URL https://github.com/baichuan-inc/Baichuan-7B.

XVERSE Technology Inc. XVERSE-13B: A multilingual large language model developed by XVERSE Technology Inc., 2023b. URL https://github.com/xverse-ai/XVERSE-13B.

InternLM Team. InternLM: A multilingual language model with progressively enhanced capabilities, 2023. URL https://github.com/InternLM/InternLM.

Shantanu Jain. tiktoken: A fast BPE tokeniser for use with OpenAI's models, 2022. URL https://github.com/openai/tiktoken/.

Yunjie Ji, Yong Deng, Yan Gong, Yiping Peng, Qiang Niu, Lei Zhang, Baochang Ma, and Xiangang Li. Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases. *arXiv preprint arXiv:2303.14742*, 2023.

Zixuan Jiang, Jiaqi Gu, Hanqing Zhu, and David Z. Pan. Pre-RMSNorm and Pre-CRMSNorm transformers: Equivalent and efficient pre-LN transformers. *CoRR*, abs/2305.14858, 2023. doi: 10.48550/arXiv.2305.14858. URL https://doi.org/10.48550/arXiv.2305.14858.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://doi.org/10.1162/tacl_a_00276.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

LangChain, Inc. LangChain: Building applications with LLMs through composability, 2023. URL https://python.langchain.com/.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022.

Chenliang Li, Hehong Chen, Ming Yan, Weizhou Shen, Haiyang Xu, Zhikai Wu, Zhicheng Zhang, Wenmeng Zhou, Yingda Chen, Chen Cheng, et al. ModelScope-Agent: Building your customizable agent system with open-source large language models. *arXiv preprint arXiv:2309.00986*, 2023a.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large scale language model society. *arXiv preprint arXiv:2303.17760*, 2023b.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: Measuring massive multitask language understanding in Chinese. *arXiv preprint arXiv:2306.09212*, 2023c.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy V, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Moustafa-Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. StarCoder: May the source be with you! *CoRR*, abs/2305.06161, 2023d. doi: 10.48550/arXiv.2305.06161. URL https://doi.org/10.48550/arXiv.2305.06161.

Yujia Li, David H. Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *CoRR*, abs/2203.07814, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Chenxiao Liu and Xiaojun Wan. CodeQA: A question answering dataset for source code comprehension. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pp. 2618–2632. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-emnlp.223. URL https://doi.org/10.18653/v1/2021.findings-emnlp.223.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023a.

Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. WebGLM: Towards an efficient web-enhanced question answering system with human preferences. *arXiv preprint arXiv:2306.07906*, 2023b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yue Liu, Thanh Le-Cong, Ratnadira Widyasari, Chakkrit Tantithamthavorn, Li Li, Xuan-Bach Dinh Le, and David Lo. Refining ChatGPT-generated code: Characterizing and mitigating code quality issues. *CoRR*, abs/2307.12596, 2023c. doi: 10.48550/arXiv.2307.12596. URL https://doi.org/10.48550/arXiv.2307.12596.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The Flan collection: Designing data and methods for effective instruction tuning. *arXiv preprint arXiv:2301.13688*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. #InsTag: Instruction tagging for analyzing supervised fine-tuning of large language models. *CoRR*, abs/2308.07074, 2023. doi: 10.48550/arXiv.2308.07074. URL https://doi.org/10.48550/arXiv.2308.07074.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. WizardMath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023a.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023b.

Mosaic ML. MPT-30B: Raising the bar for open-source foundation models, 2023. URL https://www.mosaicml.com/blog/mpt-30b.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*, 2022.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro von Werra, and Shayne Longpre. OctoPack: Instruction tuning code large language models. *CoRR*, abs/2308.07124, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Maxwell Nye, Anders Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *ArXiv*, abs/2112.00114, 2021.

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

OpenAI. ChatML, 2022. URL https://github.com/openai/openai-python/blob/e389823ba013a24b4c32ce38fa0bd87e6bccae94/chatml.md.

OpenAI. GPT4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenCompass Team. OpenCompass: A universal evaluation platform for foundation models, 2023. URL https://opencompass.org.cn/leaderboard-llm.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1144. URL https://doi.org/10.18653/v1/p16-1144.

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

Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023b.

Qwen Team, Alibaba Cloud. Evaluation benchmark for code intepreter, 2023a. URL https://github.com/QwenLM/Qwen-Agent/tree/main/benchmark.

Qwen Team, Alibaba Cloud. Evaluation benchmark for tool usage throught ReAct prompting, 2023b. URL https://github.com/QwenLM/Qwen-7B/tree/main/eval.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=1ikK0kHjvj.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code Llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. SocialIQA: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728, 2019. URL http://arxiv.org/abs/1904.09728.

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.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

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

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

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace. *arXiv preprint arXiv:2303.17580*, 2023.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Qingyi Si, Tong Wang, Naibin Gu, Rui Liu, and Zheng Lin. Alpaca-CoT: An instruction-tuning platform with unified interface of instruction collection, parameter-efficient methods, and large language models, 2023. URL https://github.com/PhoebusSi/alpaca-CoT.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*, 2023.

Stability AI. StableBeluga2, 2023. URL https://huggingface.co/stabilityai/StableBeluga2.

Jianlin Su. Improving transformer: Length extrapolation ability and position robustness, 2023a. URL https://spaces.ac.cn/archives/9444.

Jianlin Su. The magical effect of the Bias term: RoPE + Bias = better length extrapolation, 2023b. URL https://spaces.ac.cn/archives/9577.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Tianxiang Sun, Xiaotian Zhang, Zhengfu He, Peng Li, Qinyuan Cheng, Hang Yan, Xiangyang Liu, Yunfan Shao, Qiong Tang, Xingjian Zhao, Ke Chen, Yining Zheng, Zhejian Zhou, Ruixiao Li, Jun Zhan, Yunhua Zhou, Linyang Li, Xiaogui Yang, Lingling Wu, Zhangyue Yin, Xuanjing Huang, and Xipeng Qiu. MOSS: Training conversational language models from synthetic data, 2023a.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv preprint arXiv:2305.03047*, 2023b.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

Marc Szafraniec, Baptiste Rozière, Hugh Leather, Patrick Labatut, François Charton, and Gabriel Synnaeve. Code translation with compiler representations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=XomEU3eNeSQ.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4149–4158. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1421. URL https://doi.org/10.18653/v1/n19-1421.

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, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science, 2022.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Agüera y Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. LaMDA: Language models for dialog applications. *CoRR*, abs/2201.08239, 2022. URL https://arxiv.org/abs/2201.08239.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

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 Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/arXiv.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *Conference on Empirical Methods in Natural Language Processing*, 2017. URL https://api.semanticscholar.org/CorpusID:910689.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. How far can camels go? Exploring the state of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*, 2023b.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 13484–13508. Association for Computational Linguistics, 2023c. doi: 10.18653/v1/2023.acl-long.754. URL https://doi.org/10.18653/v1/2023.acl-long.754.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*, 2021.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. CodeT5+: Open code large language models for code understanding and generation. *CoRR*, abs/2305.07922, 2023d. doi: 10.48550/arXiv.2305.07922. URL https://doi.org/10.48550/arXiv.2305.07922.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022a. URL https://openreview.net/forum?id=gEZrGCozdqR.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022b. URL https://api.semanticscholar.org/CorpusID:249674500.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022c.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. HuggingFace's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. ExpertPrompting: Instructing large language models to be distinguished experts. *arXiv preprint arXiv:2305.14688*, 2023a.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. WizardLM: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023b.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*, 2023c.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*, 2023d.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, 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. Baichuan 2: Open large-scale language models. Technical report, Baichuan Inc., 2023. URL https://cdn.baichuan-ai.com/paper/Baichuan2-technical-report.pdf.

Jianxin Yang. Firefly. https://github.com/yangjianxin1/Firefly, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. mPLUG-Owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models, 2023.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023a.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*, 2023b.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. RRHF: Rank responses to align language models with human feedback without tears, 2023c.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. GLM-130B: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

Fengji Zhang, Bei Chen, Yue Zhang, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. RepoCoder: Repository-level code completion through iterative retrieval and generation. *CoRR*, abs/2303.12570, 2023a. doi: 10.48550/arXiv.2303.12570. URL https://doi.org/10.48550/arXiv.2303.12570.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. OPT: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the performance of large language models on GAOKAO benchmark. *CoRR*, abs/2305.12474, 2023b. doi: 10.48550/arXiv.2305.12474. URL https://doi.org/10.48550/arXiv.2305.12474.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. CodeGeeX: A pre-trained model for code generation with multilingual evaluations on humaneval-x. *CoRR*, abs/2303.17568, 2023. doi: 10.48550/arXiv.2303.17568. URL https://doi.org/10.48550/arXiv.2303.17568.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. *CoRR*, abs/2304.06364, 2023a. doi: 10.48550/arXiv.2304.06364. URL https://doi.org/10.48550/arXiv.2304.06364.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *arXiv preprint arXiv:2305.10250*, 2023b.

Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Huai hsin Chi. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625, 2022.

A Appendix

A.1 More Training Details

A.1.1 Data Format for Qwen-Chat

Different from conventional pretraining based on autoregressive next-token prediction, despite using a similar training task, there should be a specially design data format for SFT and RLHF to build a conversational AI assistant model. Common formats include "human-assistant" and ChatML formats. As to our knowledge, the earliest example of the human-assistant format comes from Anthropic (Bai et al., 2022b), which adds a special phrase "`\n\nhuman: `" in front of the user input and "`\n\nassistant: `" in front of the assistant response. It is easy for the base language model to transfer to the pattern of conversational AI. However, as the specific phrases are common words, it might be hard for the model to disambiguate from these words in other contexts.

Instead, we turned to the ChatML format proposed by OpenAI.[5] This format allows the use of special tokens, i.e., "`<im_start>`" and "`<im_end>`", that do not appear in pretraining, and thus resolve the aforementioned problem. We demonstrate an example of the format below.

ChatML Format

```
<|im_start|>system
You are a helpful assistant.<|im_end|>
<|im_start|>user
Hello!<|im_end|>
<|im_start|>assistant
Hello! How can I assist you today?<|im_end|>
```

A.2 Evaluation

A.2.1 Automatic Evaluation

To provide a whole picture of the performance of our model series Qwen, here in this section we illustrate the detailed performance of our models as well as the baselines in the comprehensive benchmark evaluation proposed by OpenCompass Team (2023). We report the results in multiple tables based on the officially provided categories, including examination, language, knowledge, understanding, and reasoning. In terms of the performance of the baseline models, we report the higher results between the reported ones and those on the leaderboard.

Examination Here we evaluate the models on a series of datasets relevant to the examination. The datasets include

- **MMLU** (Hendrycks et al., 2020) Massive Multi-task Language Understanding is designed for measuring language understanding capabilities. We report 5-shot results.
- **C-Eval** (Huang et al., 2023) C-Eval is a Chinese evaluation dataset spanning 52 diverse disciplines. We report 5-shot results.
- **CMMLU** (Li et al., 2023c) CMMLU is designed for assessing language understanding capabilities in Chinese. We report 5-shot results.
- **AGIEval** (Zhong et al., 2023a) This is a benchmark consisting of human-centric examinations, including college entrance exams, law school admission tests, math competitions, and lawyer qualification tests. We report zero-shot results.
- **Gaokao-Bench** (Zhang et al., 2023b) This is a benchmark with Gaokao (Chinese college-entrance examination) questions. We report zero-shot results.
- **ARC** (Clark et al., 2018) ARC is a dataset consisting of grade-school level, multiple-choice science questions. It includes an easy set and a challenge set, which are referred by ARC-e and ARC-c. We report zero-shot results.

Table 13: **Results on MMLU**. All are tested with five-shot accuracy. We provide the reported results of the other models for comparison.

Model	Params	Average	STEM	Social Sciences	Humanities	Others
MPT	7B	26.8	25.3	27.1	26.7	28.2
	30B	46.9	39.0	52.8	44.5	52.9
Falcon	7B	26.2	26.2	24.7	26.4	27.4
	40B	55.4	45.5	65.4	49.3	65.0
ChatGLM2	6B	47.9	41.2	54.4	43.7	54.5
	12B	56.2	48.2	65.1	52.6	60.9
InternLM	7B	51.0	-	-	-	-
Baichuan2	7B	54.2	-	-	-	-
	13B	59.2	-	-	-	-
XVERSE	13B	55.1	44.5	64.4	50.5	62.9
LLaMA	7B	35.1	30.5	38.3	34.0	38.1
	13B	46.9	35.8	53.8	45.0	53.3
	33B	57.8	46.0	66.7	55.8	63.4
	65B	63.4	51.7	72.9	61.8	67.4
LLAMA 2	7B	45.3	36.4	51.2	42.9	52.2
	13B	54.8	44.1	62.6	52.8	61.1
	34B	62.6	52.1	71.8	59.4	69.2
	70B	68.9	58.0	80.3	65.0	74.6
QWEN	1.8B	44.6	39.6	50.0	40.4	51.0
	7B	58.2	50.2	68.6	52.5	64.9
	14B	**66.3**	**59.4**	**76.2**	**60.9**	**71.8**

Table 14: **Leaderboard results of C-Eval**. We include the results of both proprietary models and open-source models. Note that there are a number of models on the leaderboard with very few details, in terms of proprietary models, we only report the results of GPT-3.5, GPT-4, InternLM and ChatGLM2.

Model	Params	Avg.	Avg. (Hard)	STEM	Social Sciences	Humanities	Others
Proprietary models							
GPT-3.5	-	54.4	41.4	52.9	61.8	50.9	53.6
GPT-4	-	68.7	**54.9**	**67.1**	77.6	64.5	67.8
InternLM	123B	68.8	50.0	63.5	81.4	72.7	63.0
ChatGLM2	-	**71.1**	50.0	64.4	**81.6**	**73.7**	**71.3**
Open-source models							
ChatGLM2	6B	51.7	37.1	48.6	60.5	51.3	49.8
InternLM	7B	52.8	37.1	48.0	67.4	55.4	45.8
Baichuan2	7B	54.0	-	-	-	-	-
	13B	58.1	-	-	-	-	-
XVERSE	13B	54.7	33.5	45.6	66.2	58.3	56.9
QWEN	1.8B	54.7	41.8	50.8	69.9	56.3	46.2
	7B	63.5	46.4	57.7	78.1	66.6	57.8
	14B	**72.1**	**53.7**	**65.7**	**85.4**	**75.3**	**68.4**

In terms of MMLU, we report the detailed results in Table 13. In terms of C-Eval, we report the results in Table 14. For the rest of the datasets, we report the results in Table 15. Note that AGIEval includes

[5]https://github.com/openai/openai-python/blob/main/chatml.md

Table 15: **Results on the other datasets of examination**. Specifically, we report the results on CMMLU, AGIEval, ARC-e, and ARC-c.

Model	Params	CMMLU	AGIEval	Gaokao-Bench	ARC-e	ARC-c
MPT	7B	25.9	21.3	19.8	70.2	42.6
Falcon	7B	-	-	-	70.0	42.4
ChatGLM2	6B	49.3	39.0	46.4	73.0	61.0
InternLM	7B	51.8	36.9	43.0	78.7	69.5
	20B	59.0	44.6	45.5	86.1	81.7
Baichuan2	7B	57.1	42.7	47.5	54.7	32.5
	13B	62.0	48.2	54.3	61.9	38.0
LLaMA	7B	26.8	20.6	21.3	72.8	47.6
	13B	31.5	22.0	20.4	74.8	52.7
	33B	36.0	33.5	18.9	80.0	67.5
	65B	40.6	33.9	19.1	80.6	69.5
LLAMA 2	7B	31.8	21.8	18.9	75.2	45.9
	13B	38.4	30.9	18.2	77.3	60.3
	70B	53.6	40.2	23.3	85.9	78.3
StableBeluga2	70B	51.8	41.6	40.9	91.2	86.1
QWEN	1.8B	49.3	36.9	44.9	71.6	53.2
	7B	62.2	45.8	52.5	84.0	75.3
	14B	**71.0**	**52.3**	**61.9**	**90.3**	**84.4**

the parts of Chinese and English, while LLAMA 2 only reported the results in the English part, so we use the results on OpenCompass. Additionally, while CMMLU, AGIEval, and Gaokao-Bench are related to Chinese, and MPT, Falcon, and the LLaMA series were not optimized for Chinese, these models achieved low performance on the datasets.

Knowledge and Understanding Here we evaluate the models on a series of datasets relevant to knowledge and natural language understanding. The datasets include

- **BoolQ** (Clark et al., 2019) This is a QA dataset, where the questions are about passages of Wikipedia, and the model should answer yes or no to the given possible answer. We report zero-shot results.
- **CommonsenseQA** (Talmor et al., 2019) This is a dataset of multiple-choice question answering that asseses the understanding of commonsense knowledge. We report 8-shot results.
- **NaturalQuestions** (Kwiatkowski et al., 2019) It is a dataset of QA where the questions are from users and the answers are verified by experts. We report zero-shot results.
- **LAMBADA** (Paperno et al., 2016) This is dataset to evaluate language understanding by word prediction. It consists of passages related to human subjects. We report zero-shot results.

We report the results in Table 16.

Reasoning We report the evaluation results on the datasets concerning reasoning, focusing on natural language reasoning. For the others, such as mathematics and coding, as we have illustrated detailed results, here we do not report those results repeatedly. The datasets for evaluation include:

- **HellaSwag** (Zellers et al., 2019) This is a commonsense natural language inference (NLI) dataset, where the questions are easy for humans but struggling for previous language models. We report zero-shot results.
- **PIQA** (Bisk et al., 2020) This is an NLI dataset assessing the physical knowledge. We report zero-shot results.

Table 16: **Results on the datasets concerning knowledge and understanding.** Specifically, we report the results on BoolQ, CommonsenseQA, NaturalQuestions, and LAMBADA.

Model	Params	BoolQ	CommonsenseQA	NaturalQuestions	LAMBADA
MPT	7B	75.0	61.8	11.6	70.0
Falcon	7B	67.5	20.8	15.7	-
ChatGLM2	6B	79.0	65.4	9.7	54.3
InternLM	7B	64.1	59.8	8.9	67.0
	20B	87.5	70.6	25.2	71.8
XVERSE	13B	64.2	62.2	0.3	48.2
Baichuan2	7B	63.2	63.0	9.4	73.3
	13B	67.0	65.6	16.3	74.0
LLaMA	7B	76.5	64.9	16.8	73.3
	13B	78.7	67.4	20.2	75.2
	33B	84.4	72.5	30.9	77.2
	65B	86.6	74.1	33.4	77.7
LLAMA 2	7B	77.4	66.5	19.1	73.3
	13B	82.4	67.3	**24.9**	**76.5**
	70B	87.7	78.5	34.2	78.9
StableBeluga2	70B	89.4	72.6	25.1	71.3
QWEN	1.8B	68.0	60.1	3.2	58.4
	7B	76.4	66.8	17.4	67.9
	14B	**86.2**	**70.3**	23.9	71.1

Table 17: **Results on the datasets related to natural language reasoning.** Specifically, we report the results on HellaSwag, PIQA, SIQA, and OCNLI.

Model	Params	HellaSwag	PIQA	SIQA	OCNLI
MPT	7B	76.4	**80.6**	48.5	30.0
Falcon	7B	74.1	76.7	47.2	-
ChatGLM2	6B	57.0	69.6	64.3	33.1
InternLM	7B	70.6	77.9	60.5	37.5
	20B	78.1	80.3	72.8	42.5
Baichuan2	7B	67.0	76.2	44.4	30.3
	13B	70.8	78.1	44.3	30.0
LLaMA	7B	76.1	79.8	48.9	33.6
	13B	79.2	80.1	52.5	32.1
	33B	82.8	82.3	57.8	30.7
	65B	84.2	82.8	61.2	44.9
LLAMA 2	7B	77.2	78.8	48.5	32.1
	13B	**80.7**	80.5	54.8	34.1
	70B	85.3	82.8	64.8	46.5
StableBeluga2	70B	84.1	83.3	78.1	48.3
QWEN	1.8B	56.7	73.3	56.1	39.0
	7B	75.1	77.9	69.9	47.4
	14B	80.2	79.9	**77.9**	**57.9**

- **SIQA** (Sap et al., 2019) This is an NLI dataset evaluating social commonsense intelligence. We report zero-shot results.
- **OCNLI** (Hu et al., 2020) This is an NLI dataset focusing on Chinese. We report zero-shot results.

We report the results in Table 17.

A.2.2 HUMAN EVALUATION

In this section, we demonstrate the cases of human analysis. In our self-constructed evaluation dataset, the instructions are either manually written data or manual revised from public datasets, such as CLiB[6], C-Eval (Huang et al., 2023), FacTool (Chern et al., 2023b), LeetCode[7]), etc.

In terms of each case, we demonstrate the responses and Elo ratings[8] of all models for comparison. Specifically, as the data in our human evaluation are in Chinese, we also provide their translations in English.

Knowledge

北京大学的前身是什么学校?
What was the predecessor of Peking University?

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	北京大学的前身是京师大学堂，它创建于1898年，是中国近代第一所国立大学，也是最早以“大学”命名的高等学府。 The predecessor of Peking University was the Imperial University of Peking, established in 1898. It was the first national university in modern China and also the earliest higher education institution to be named a “university”.	1037
Qwen-14B-Chat (SFT)	北京大学的前身是京师大学堂，创建于1898年，是中国近代第一所国立大学。 The predecessor of Peking University is the Imperial University of Peking, founded in 1898 and the first national university in modern China.	1011
Qwen-14B-Chat (RLHF)	北京大学的前身是京师大学堂，创建于1898年，是中国近代第一所国立综合性大学。京师大学堂最初设立的学科有经学、文学、史学、哲学、法律、教育、农学、工学、商学、医学等，后来逐渐扩大到社会科学、自然科学、工程技术、艺术等多个领域。1912年，京师大学堂改名为北京大学，成为中国最高学府之一，也是中国现代高等教育的发源地之一。 Peking University, originally known as the Imperial University of Peking, was founded in 1898 and is the first comprehensive national university in modern China. The initial disciplines established at the Imperial University of Peking included classics, literature, history, philosophy, law, education, agriculture, engineering, business, medicine, etc., gradually expanding to encompass social sciences, natural sciences, engineering and technology, arts, and other fields. In 1912, the Imperial University of Peking changed its name to Peking University, becoming one of the top universities in China and also one of the birthplaces of modern higher education in China.	1090
GPT-3.5	北京大学的前身是国立北京大学，建于1898年，是中国近代第一所国立大学。其成立初衷是为了培养国家的政治、经济、军事和外交人才，为中国的现代化进程做出贡献。 The predecessor of Peking University was National Peking University, established in 1898. It is the first national university in modern China. Its original purpose was cultivating talents in politics, economics, military, and diplomacy and contributing to China’s modernization process.	910

[6]https://github.com/jeinlee1991/chinese-llm-benchmark
[7]https://leetcode.cn/
[8]https://en.wikipedia.org/wiki/Elo_rating_system

GPT-4	北京大学的前身是京师大学堂。 The predecessor of Peking University is the Imperial University of Peking.	955

Knowledge

世界上现存最高的木塔是中国什么塔?
What is the tallest wooden tower in the world that still exists in China?

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	世界上现存最高的木塔是中国山西应县木塔。 The tallest existing wooden tower in the world is the Yinxian Wooden Tower in Shanxi, China.	987
Qwen-14B-Chat (SFT)	现存最高的木塔是中国山西应县木塔，全名为佛宫寺释迦塔。应县木塔位于山西省朔州市应县县城内西北角的佛宫寺内，建于辽代清宁二年（1056年），金明昌六年（1195年）增修完毕，是中国现存最高最古的一座木构塔式建筑，也是世界现存最高的木结构建筑。 The tallest existing wooden tower in the world is the Yinxian Wooden Tower, also known as the Shijia Tower of the Fogong Temple. It is inside the Fogong Temple in Yinxian, Shanxi Province, China. The tower was originally built in the second year of the Qingning era of the Liao Dynasty (1056 AD) and completed its expansion in the sixth year of the Jin Dynasty's Mingchang era (1195 AD). It is the tallest and oldest existing wooden tower in China and the tallest wooden structure in the world.	1055
Qwen-14B-Chat (RLHF)	世界上现存最高的木塔是中国的应县木塔，位于山西省应县。该塔建于公元1056年，高67.31米，共有5层，是中国古代建筑技术的杰出代表之一。应县木塔的建筑风格独特，采用了许多古代建筑技术，如榫卯结构、斗拱、悬挑等，具有很高的艺术价值和历史价值。应县木塔也是中国文化遗产保护的重点之一，被列为全国重点文物保护单位和世界文化遗产。 The tallest existing wooden tower in the world is the Yinxian Wooden Tower in Yinxian, Shanxi Province, China. It was built in the year 1056 AD and stands at a height of 67.31 meters, consisting of 5 floors. This tower is an outstanding representative of ancient Chinese architectural technology. The Yinxian Wooden Tower showcases a unique architectural style and incorporates various ancient construction techniques such as mortise and tenon joints, bracket sets, and cantilever beams, possessing high artistic and historical value. It is also a key cultural heritage site in China, listed as a national key cultural relic protection unit and a UNESCO World Cultural Heritage site.	1060
GPT-3.5	世界上现存最高的木塔是中国的普陀山普济寺塔，也被称为古佛顶宝塔。该塔建于1056年，高约70.28米。 The tallest existing wooden tower in the world is the Putuoshan Pujisi Pagoda, also known as the Ancient Baoding Pagoda, located in Putuoshan, China. It was built in the year 1056 AD and stands at a height of approximately 70.28 meters.	864
GPT-4	世界上现存最高的木塔是中国的应县木塔，也被称为佛宫寺大佛塔。	1040

The tallest wooden pagoda currently in existence in the world is the Yingxian Wooden Pagoda in China, also known as the Pagoda of Fogong Temple.

Language Understanding

吃葡萄不吐葡萄皮的拼音怎么写?
What is the Pingyin for “吃葡萄不吐葡萄皮”?

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	吃葡萄不吐葡萄皮的拼音是：“chī pú bù tǔ pú pí”。 To eat grapes without spitting out the grape skins, the pinyin is “chī pú bù tǔ pú pí”.	914
Qwen-14B-Chat (SFT)	拼音：吃葡萄不吐葡萄皮“chī pú táo bù tǔ pú táo pí” Pingyin: To eat grapes without spitting out the grape skins, ‘chī pú bù tǔ pú pí”.	1049
Qwen-14B-Chat (RLHF)	吃葡萄不吐葡萄皮的拼音是：chī pú táo bù tǔ pú táo pí。拼音是一种用拉丁字母表示汉字发音的方法，它可以帮助人们学习和记忆汉字的发音。在这个拼音中，“吃”表示“chī”，“葡萄”表示“pú táo”，“不”表示“bù”，“吐”表示“tǔ”，“葡萄皮”表示“pú táo pí”。 The pinyin for ’吃葡萄不吐葡萄皮’ is “chī pú táo bù tǔ pú táo pí”. Pinyin is a method of representing the pronunciation of Chinese characters using Latin letters, which can help people learn and remember the pronunciation of Chinese characters. In this pinyin, “吃” is represented as ’chī”, “葡萄” is represented as “pú táo”, “不” is represented as “bù”, “吐” is represented as “tǔ”, and “葡萄皮” is represented as “pú táo pí”.	1068
GPT-3.5	chī pú táo bú tǔ pú táo pí de pīn yīn zěn me xiě? chī pú táo bú tǔ pú táo pí de pīn yīn zěn me xiě?	946
GPT-4	chī pú táo bù tǔ pú táo pí chī pú táo bù tǔ pú táo pí	1040

Language Understanding

父亲：儿子，你高中毕业了，现在打算上哪个大学?
儿子：爸爸，我已经准备好了申请资料，想申请一个好的本科专业。
父亲：那你已经想好申请哪些大学了吗?
儿子：是的，我计划申请几所排名靠前的大学。我已经做了很多研究，准备在学业、社交和职业规划等方面全面提升自己。
父亲：很好！你一直都很努力，爸爸妈妈一直都很自豪。上大学是一个新的起点，你需要面对更多的挑战和机会。你有什么打算?
儿子：我计划尽早适应大学生活，参加各种社团和课外活动，拓宽自己的视野和人际圈。同时，我会认真学习专业知识和技能，努力取得好成绩，争取获得奖学金和其他荣誉。
父亲：非常好！你的计划很明智，爸爸相信你一定会有一个充实而有意义的大学生活。爸爸会一直支持你的决定，希望你取得好成绩，为你的未来打下坚实的基础。
儿子：谢谢爸爸！我会加倍努力，不辜负您对我的期望。
这段对话中，儿子打算上大学后有哪些计划和打算?

Father: Son, you have graduated from high school. Which university are you planning to attend now?
Son: Dad, I have already prepared my application materials, and I want to apply for an excellent undergraduate program.
Father: Have you already decided which universities you want to apply to?
Son: Yes, I plan to apply to several top-ranked universities. I have done a lot of research, and I am ready to improve myself in academics, social life, and career planning.

Father: That's great! You have always been hardworking and your mom and I are proud of you. Going to college is a new starting point, and you will face more challenges and opportunities. What are your plans?
Son: I plan to adapt to college life as soon as possible and participate in various clubs and extracurricular activities to broaden my horizons and social circle. At the same time, I will study my major knowledge and skills seriously, strive for good grades, and try to win scholarships and other honors.
Father: Very good! Your plans are wise. Dad believes that you will have a fulfilling and meaningful college life. Dad will always support your decisions and hopes you achieve good results, laying a solid foundation for your future.
Son: Thank you, Dad! I will work even harder and not let your expectations down.
What are the son's plans and intentions after going to college in this conversation?

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	儿子打算尽早适应大学生活，参加各种社团和课外活动，拓宽自己的视野和人际圈。同时，他会认真学习专业知识和技能，努力取得好成绩，争取获得奖学金和其他荣誉。 My son plans to adapt to university life as soon as possible, participate in various clubs and extracurricular activities, and expand his horizons and social circle. At the same time, he will study professional knowledge and skills seriously, work hard to achieve good grades, and strive to obtain scholarships and other honors.	973
Qwen-14B-Chat (SFT)	儿子打算在学业、社交和职业规划等方面全面提升自己，尽早适应大学生活，参加各种社团和课外活动，拓宽自己的视野和人际圈。同时，他会认真学习专业知识和技能，努力取得好成绩，争取获得奖学金和其他荣誉。 The son plans to comprehensively improve himself in his academic, social, and career planning aspects to adapt to university life as early as possible. He will participate in various clubs and extracurricular activities to broaden his horizons and expand his social circle. At the same time, he will diligently study professional knowledge and skills, strive for good grades, and obtain scholarships and other honors.	1049
Qwen-14B-Chat (RLHF)	儿子打算在大学生活中参加各种社团和课外活动，拓宽自己的视野和人际圈。同时，他会认真学习专业知识和技能，努力取得好成绩，争取获得奖学金和其他荣誉。他的目标是尽早适应大学生活，为自己的未来打下坚实的基础。 My son plans to participate in various clubs and extracurricular activities during his college life, in order to broaden his horizons and expand his social circle. At the same time, he will study professional knowledge and skills seriously, strive for good grades, and aim to obtain scholarships and other honors. His goal is to adapt to college life as soon as possible and lay a solid foundation for his future.	975
GPT-3.5	儿子计划适应大学生活，参加各种社团和课外活动，拓宽自己的视野和人际圈。同时，他还计划认真学习专业知识和技能，努力取得好成绩，争取获得奖学金和其他荣誉。他希望在大学打下坚实的基础，为自己的未来打好基础。 The son plans to adapt to college life participate in various clubs and extracurricular activities to broaden his horizons and social circle. At the same time, he plans to study professional knowledge and skills seriously, strive for good grades, and obtain scholarships and other honors. He hopes to lay a solid foundation in college and build a good foundation for his future.	976

GPT-4	儿子的大学计划和打算包括： 1. 尽早适应大学生活。 2. 参加各种社团和课外活动，以拓宽自己的视野和人际圈。 3. 认真学习专业知识和技能，努力取得好成绩。 4. 争取获得奖学金和其他荣誉。 5. 为未来打下坚实的基础。 My son's university plans and aspirations include: 1. Adjusting to university life as soon as possible. 2. Joining various clubs and extracurricular activities to broaden his horizons and social circle. 3. Studying professional knowledge and skills seriously and striving for excellent grades. 4. Striving to obtain scholarships and other honors. 5. Building a solid foundation for the future.	1048

Creative Writing

请编写一个以超市为背景的，内容涉及到水果误会的笑话。
Please write a joke set in a supermarket involving a fruit misunderstanding.

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	有一天，小明去超市买水果，他买了一篮子苹果。当他回到家，他发现苹果都是红色的，他想：这些苹果一定很好吃。但是，当他开始吃的时候，他发现这些苹果都是酸的。他很惊讶，因为他买的是甜苹果。他去超市问，店员说："对不起，我们没有甜苹果，只有红色的苹果。"小明很生气，他想：我买的是甜苹果，为什么给我红色的苹果呢？他回去告诉了妈妈，妈妈笑着说："你买的是红苹果... [[The original content is long and the follow up is truncated]] One day, Xiaoming went to the supermarket to buy fruits. He bought a basket of apples. When he got home, he found that all the apples were red. He thought to himself: These apples must be very delicious. However, when he started eating them, he found that these apples were sour. He was surprised because he bought sweet apples. He went to the supermarket to ask, and the shopkeeper said, "I'm sorry, we don't have sweet apples, only red apples." Xiaoming was angry and thought to himself: I bought sweet apples, why did they give me red apples? He went back and told his mother, and his mother smiled and said, "You bought red delicious apples.... [[The original content is long and the follow up is truncated]]	849

Qwen-14B-Chat (SFT)	一天，一位顾客走进超市，对店员说：“我要买一个香蕉。” 店员回答：“好的，先生，我们这里有各种各样的香蕉，您想要哪一种？” 顾客回答：“我要一个黄色的香蕉。” 店员有些疑惑，但还是给他拿了一个黄色的香蕉。 顾客又说：“我要一个弯曲的香蕉。” 店员更加疑惑了，但他还是给他拿了一个弯曲的香蕉。 最后，顾客说：“我要一个会说话的香蕉。” 店员忍不住笑了，回答：“先生，我们这里没有会说话的香蕉，那只是个误会。” 顾客也笑了，回答：“我知道，我只是想和你开个玩笑。” One day, a customer walked into a supermarket and said to the shop assistant, “I want to buy a banana.” The assistant replied, “Okay, sir, we have all kinds of bananas here. Which one would you like?” The customer answered, “I want a yellow banana.” The assistant was a little confused but still gave him a yellow banana. The customer then said, “I want a curved banana.” The assistant was even more confused, but he still gave him a curved banana. Finally, the customer said, “I want a talking banana.” The assistant couldn’t help but laugh and replied, ”Sir, we don’t have bananas that can talk here, that’s just a misunderstanding.” The customer also laughed and replied, “I know, I was just joking with you.”	980

Qwen-14B-Chat (RLHF)	有一天，小明去超市买水果。他挑选了一篮子新鲜的苹果和香蕉，然后走向收银台。 收银员小王看到小明手中的水果，突然大声喊道：“哇，你的水果好新鲜啊！你是从哪里买的？” 小明愣了一下，心想：“难道我买水果的技巧被发现了？”于是他得意地回答：“我刚刚从果园里摘的，保证新鲜！” 小王听了，忍不住笑了起来：“你可真逗，这是超市里的水果，不是你家果园里的。” 小明听了，顿时傻眼了。他尴尬地解释道：“啊，对不起，我太紧张了，以为这是我的水果。” 小王笑着安慰道：“没事没事，我们都搞错了。不过，你的水果确实很好，我们超市里的水果也都是新鲜的。” 小明听了，松了一口气，心情也好了起来。他付了钱，提着水果离开了超市，心里想着：“下次再买水果，一定要更加小心，不要闹出笑话了！” One day, Xiao Ming went to the supermarket to buy fruits. He picked a basket of fresh apples and bananas, and then headed towards the cashier. The cashier, Xiao Wang, saw the fruits in Xiao Ming’s hands and suddenly shouted, ”Wow, your fruits are so fresh! Where did you buy them from?” Xiao Ming was stunned for a moment and thought to himself, ”Could it be that my fruit-buying skills have been discovered?” So he proudly replied, ”I just picked them from the orchard, guaranteed fresh!” Upon hearing this, Xiao Wang couldn’t help but laugh and said, “You’re really funny. These are fruits from the supermarket, not from your own orchard.” Xiao Ming, upon listening, was immediately dumbfounded. He awkwardly explained, “Oh, I’m sorry, I was too nervous, and I thought these were my fruits.” Xiao Wang smiled and comforted him, saying, “It’s alright, it’s alright. We both made a mistake. However, your fruits are indeed very good, and the fruits in our supermarket are also fresh.” Upon hearing this, Xiao Ming breathed a sigh of relief, and his mood improved. He paid the money, carried the fruits, and left the supermarket, thinking, “Next time I buy fruits, I must be more careful and avoid making such a funny mistake!”	986
GPT-3.5	在超市的水果区，小明看到了一盒红苹果和一盒绿苹果。他拿着两盒苹果走到收银台前排队结账。 收银员看着小明手里的两盒苹果说：“这位小朋友，你知道红苹果和绿苹果有什么区别吗？” 小明想了一下，然后笑着回答：“当然知道了！红苹果是情侣，绿苹果是单身狗。” 收银员被小明的回答逗乐了，忍不住笑了出来。	964

	In the supermarket's fruit section, Xiaoming saw a box of red apples and a box of green apples. He held the two boxes of apples and walked to the cash register to queue up for payment. The cashier looked at the two boxes of apples in Xiaoming's hands and said, "Hey, little friend, do you know the difference between red apples and green apples?" Xiaoming thought momentarily and then smiled, "Of course, I know! Red apples are couples, and green apples are single dogs." Xiaoming's answer amused the cashier, and couldn't help but laugh.	
GPT-4	"一个非常善良的男孩，在超市里发现了一个老太太在苹果堆里挑选苹果。他看到老太太似乎找不到满意的苹果，就走上前去帮忙。 男孩说："老奶奶，这箱子的苹果我之前都看过了，里面没有一个好的。但是我知道柜台上有一个我保留的极好的苹果，我去为您拿过来。" 老太太感动地点点头，男孩兴冲冲地跑去拿苹果，只见他笑呵呵地拿过一个巨大的，艳丽的苹果放在老太太面前。 老太太疑惑地看看男孩，再看看手中的"苹果"，然后微笑着说："孩子，很感谢你的好意，但这是一个番茄。"" A very kind boy found an old lady picking apples in a pile at the supermarket. Seeing that she could not find a satisfactory apple, he approached to help. The boy said, "Granny, I've checked all the apples in this box before, and there isn't a good one. But I know I saved an excellent apple on the counter for you. Let me get it for you." The old lady nodded gratefully, and the boy eagerly ran to get the apple. He joyfully handed a huge, vibrant apple to the old lady. Confused, the old lady looked at the boy, then at the "apple" in her hand, and smiled. She said, "Child, thank you for your kindness, but this is a tomato."	1144

Mathematics

一个正方形，被分成5个相等的长方形，每个长方形的周长是60厘米，正方形的周长是多少厘米
A square is divided into five equal rectangles, each with a perimeter of 60 centimeters. What is the perimeter of the square in centimeters?

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	设正方形的边长为x，那么每个长方形的边长就是$x/5$。根据题意，我们知道每个长方形的周长是60厘米，所以有： $2(x/5) + 2(x/5) + 2(x/5) + 2(x/5) = 60$ 化简得：$x = 60/(2 + 2 + 2 + 2) = 15$ 所以正方形的周长是$4x = 4 * 15 = 60$厘米 If the side length of a square is x, then the side length of each rectangle is $x/5$. According to the question, we know that the perimeter of each rectangle is 60 centimeters, so we have: $2(x/5) + 2(x/5) + 2(x/5) + 2(x/5) = 60$ Simplifying, we get: $x = 60/(2 + 2 + 2 + 2) = 15$ So the perimeter of the square is $4x = 4 * 15 = 60$ centimeters.	976

Qwen-14B-Chat (SFT)	设正方形边长为x，长方形边长为y 根据题意，有：$x = 5y$ $2y + 2y = 60$ $4y = 60$ $y = 15$ $x = 5 * 15 = 75$ 正方形周长= $4x = 4 * 75 = 300$ Let the side length of the square be x and the length of the rectangle be y. According to the problem: $x = 5y$ $2y + 2y = 60$ $4y = 60$ $y = 15$ $x = 5 * 15 = 75$ Perimeter of the square = $4x = 4 * 75 = 300$	976
Qwen-14B-Chat (RLHF)	首先，我们设正方形的边长为x，则每个长方形的长为x，宽为$x/5$。 根据长方形的周长公式，可得每个长方形的周长为$2(x + x/5) = 12x/5$。 已知每个长方形的周长为60厘米，代入上式得: $12x/5 = 60$ 解得$x = 25$。 因此，正方形的周长为$4x = 100$厘米。 答案：正方形的周长是100厘米。 First, let's assume that the square's side length is x. Therefore, the length of each rectangle is x and the width is $x/5$. According to the formula for the perimeter of a rectangle, we can determine that the perimeter of each rectangle is $2(x + x/5) = 12x/5$. Given that the perimeter of each rectangle is 60 centimeters, we can substitute this value into the equation: $12x/5 = 60$ Solving for x, we find $x = 25$. Therefore, the perimeter of the square is $4x = 100$ centimeters. Answer: The perimeter of the square is 100 centimeters.	1139
GPT-3.5	设正方形的边长为x厘米，则每个长方形的长为x厘米，宽为$5x/4$厘米。 每个长方形的周长为$2(x + 5x/4) = 60$厘米。 化简得$13x/2 = 60$ 解方程得到$x = (60 * 2)/13 = 120/13 H 9.23$ 正方形的周长为$4x = 4 * 9.23 \approx 36.92$厘米。 If the side length of a square is x centimeters, then the length of each rectangle is x centimeters, and the width is 5x/4 centimeters. The perimeter of each rectangle is 2(x + 5x/4) = 60 centimeters. Simplifying, we get 13x/2 = 60. Solving the equation, we find x = (60 * 2) / 13 = 120 / 13 $\approx$ 9.23. The perimeter of the square is 4x = 4 * 9.23 $\approx$ 36.92 centimeters.	897
GPT-4	由题意可知，每个长方形的一条长边等于正方形的边长，设为a。设每个长方形的一条短边为b，那么有$5b = a$。根据每个长方形的周长可得到$2 * (a + b) = 60$厘米。将$5b = a$代入到长方形的周长公式中得：$2 * a + 2 * 5b = 60$，化简得$2 * a + 10b = 60$，由于$a = 5b$，所以$2 * a = 2 * 5b = 10b$ 那么有$10b + 10b = 60$，得$b = 3$厘米，$a = 5 * b = 15$厘米。 所以，正方形的周长是$4 * a = 4 * 15 = 60$厘米。	1010

	From the given problem, it can be inferred that the length of one side of each rectangle is equal to that of one side of the square, denoted as a. Let the length of one side of each rectangle be denoted as b; then, we have $5b = a$. The perimeter of each rectangle can be obtained as $2 * (a + b) = 60$ centimeters. Substituting $5b = a$ into the formula for the perimeter of the rectangle, we have $2a+25b = 60$. Simplifying it gives $2a+10b = 60$. Since $a = 5b$, it follows that $2a = 25b = 10b$. Therefore, $10b + 10b = 60$, which gives $b = 3$ centimeters and $a = 5b = 15$ centimeters. Thus, the perimeter of the square is $4a = 415 = 60$ centimeters.

Mathematics

同学们进行广播操比赛，全班正好排成相等的6行。小红排在第二行，从头数，她站在第5个位置，从后数她站在第3个位置，这个班共有多少人

The students are having a radio exercise competition, and the whole class is neatly arranged into 6 equal rows. Xiaohong is in the second row, counting from the front, she is in the 5th position. Counting from the back, she is in the 3rd position. How many people are there in this class?

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	小红前面的人数= 5 - 1 = 4，小红后面的人数= 3 - 1 = 2，全班人数= 小红前面的人数+ 小红后面的人数+ 小红自己= 4 + 2 + 1 = 7 The number of people in front of Xiaohong = 5 - 1 = 4, the number of people behind Xiaohong = 3 - 1 = 2, the total number of people in the class = the number of people in front of Xiaohong + the number of people behind Xiaohong + Xiaohong herself = 4 + 2 + 1 = 7.	976
Qwen-14B-Chat (SFT)	5+3-1=7（人） 7×6=42（人） 这个班共有42人。 5+3-1=7 (people) 7×6=42 (people) There are a total of 42 people in this class.	976
Qwen-14B-Chat (RLHF)	小红前面有4个人，后面有2个人，所以这一行共有4+2+1=7个人。 全班共有6行，所以全班共有6*7=42个人。 There are 4 people in front of Xiaohong and 2 people behind her, so there are a total of 4+2+1=7 people in this row. There are a total of 6 rows in the class, so there are a total of 6*7=42 people in the whole class.	1139
GPT-3.5	根据题目信息，小红排在第二行第5个位置，也即第11个位置。而小红从后数是第3个位置，也即倒数第11个位置。所以这个班共有11个人。 According to the information given, Xiaohong is in the second row, fifth position, which is also the 11th position. And Xiaohong is in the third position from the end, which is also the 11th position counting from the back. So there are a total of 11 people in this class.	897
GPT-4	如果小红从头数是第5个，从后数是第3个，那么这一排就有5+3-1=7人。全班正好排成6行，所以班级里有人数为7*6=<<7*6=42>>=42人。	1010

If Xiaohong is the 5th person when counting from the front and the 3rd person when counting from the back, then there are a total of 5+3-1=7 people in this row. The whole class is perfectly arranged in 6 rows, so there are a total of 7*6=42 people in the class.

Code

python实现一个正则表达式匹配，要求能匹配到所有以139开头，以00结尾的11位数字
Implement a regular expression matching in Python, which should be able to match all 11-digit numbers that start with ”139” and end with ”00”.

Responses

Model
Qwen-7B-Chat (SFT)

Response

你可以使用Python的re模块来实现这个正则表达式匹配。以下是一个示例代码：

```
import re

def match_phone_numbers(phone_numbers):
    pattern = r'^139d9$'
    return re.findall(pattern, phone_numbers)

phone_numbers = ['13900000000', '13912345678', '13998765432', '13987654321']
print(match_phone_numbers(phone_numbers))
```

在这个代码中，我们首先定义了一个正则表达式模式'^139d9$'，这个模式表示以139开头，以00结尾，且长度为11的数字。然后我们使用re.findall函数来查找所有匹配这个模式的字符串，最后返回这些匹配的字符串。

注意，这个代码假设phone_numbers是一个包含多个字符串的列表。如果你的phone_numbers是一个字符串，你需要先将其转换为列表。
You can use the re module in Python to implement this regular expression matching. Here is an example code:

```
import re

def match_phone_numbers(phone_numbers):
    pattern = r'^139d9$'
    return re.findall(pattern, phone_numbers)

phone_numbers = ['13900000000', '13912345678', '13998765432', '13987654321']
print(match_phone_numbers(phone_numbers))
```

This code defines a regular expression pattern ^139d9$. This pattern represents a string that starts with '139', ends with '00', and has a length of 11 digits. Then, we use the re.findall function to find all the strings that match this pattern, and finally, we return these matching strings.

Note that this code assumes that phone_numbers is a list containing multiple strings. If your phone_numbers is a string, you must first convert it to a list.

Elo
941

Qwen-14B-Chat (SFT)

你可以使用Python的re模块来实现这个正则表达式匹配。以下是一个简单的例子:

```
import re

def match_phone_numbers(phone_numbers):
    pattern = r'^139d9$'
    return [phone_number for phone_number in
phone_numbers if re.match(pattern, phone_number)]

phone_numbers = ['13912345678', '13923456789', '
13934567890', '13945678901', '13956789012', '13967890123
', '13978901234', '13989012345', '13990123456', '
13900000000']
print(match_phone_numbers(phone_numbers))
```

在这个例子中，我们定义了一个正则表达式模式'^139d9$'，它匹配以139开头，后面跟着9个任意数字，最后以0结尾的字符串。然后我们使用re.match()函数来检查每个电话号码是否匹配这个模式。如果匹配，我们就将这个电话号码添加到结果列表中。

You can use the re module of Python to implement this regular expression matching. Here is a simple example:

```
import re

def match_phone_numbers(phone_numbers):
    pattern = r'^139d9$'
    return [phone_number for phone_number in
phone_numbers if re.match(pattern, phone_number)]

phone_numbers = ['13912345678', '13923456789', '
13934567890', '13945678901', '13956789012', '13967890123
', '13978901234', '13989012345', '13990123456', '
13900000000']
print(match_phone_numbers(phone_numbers))
```

In this example, we define a regular expression pattern '^139d9$', which matches a string starting with 139, followed by nine arbitrary digits and ending with 0. Then we use the re.match() function to check if each phone number matches this pattern. If it does, we add this phone number to the result list.

941

Qwen-14B-Chat (RLHF) 可以使用Python的re模块来实现正则表达式匹配。以下是一个示例代码： 1090

```
import re

pattern = r'^139d9$'

# 测试匹配
text = '13912345678'
match = re.match(pattern, text)
if match:
    print('匹配成功')
else:
    print('匹配失败')

# 测试不匹配
text = '139123456780'
match = re.match(pattern, text)
if match:
    print('匹配成功')
else:
    print('匹配失败')
```

其中，^表示匹配字符串的开头，d表示匹配数字，9表示匹配9个数字，$表示匹配字符串的结尾。将这些符号组合在一起，就可以得到一个匹配11位数字的正则表达式。

在测试匹配时，使用re.match()函数来尝试匹配文本，如果匹配成功，返回一个Match对象，否则返回None。在本例中，第一个测试匹配成功，第二个测试匹配失败。
You can use the re module in Python to implement regular expression matching. Here is an example code:

```
import re

pattern = r'^139d9$'

# Match Test
text = '13912345678'
match = re.match(pattern, text)
if match:
    print('Success')
else:
    print('Fail')

# Unmatch Test
text = '139123456780'
match = re.match(pattern, text)
if match:
    print('Success')
else:
    print('Fail')
```

Among them, ^ represents matching the beginning of a string, d represents matching a digit, 9 represents matching 9 digits, and $ represents matching the end of a string. Combining these symbols, we can obtain a regular expression that matches 11 digits.
When testing for a match, use the re.match() function to attempt to match the text. A successful match returns a Match object; otherwise, it returns None. In this example, the first test matches successfully, while the second test fails to match.

GPT-3.5

使用Python的re模块可以很方便地实现正则表达式匹配。以下是一个匹配所有以139开头，以00结尾的11位数字的示例代码：

```
import re

text = "139123456700"
pattern = r"^139d800$"

match = re.match(pattern, text)
if match:
    print("匹配成功")
else:
    print("匹配失败")
```

输出结果为：
匹配成功
代码中，首先定义了一个文本字符串text，用于测试正则表达式的匹配。接着定义了一个正则表达式pattern，其中^139表示以139开头，d8表示任意8位数字，00$表示以00结尾。然后使用re.match()函数进行匹配，如果匹配成功，则输出"匹配成功"，否则输出"匹配失败"。
The re module in Python can conveniently accomplish regular expression matching. Below is an example code that matches all 11-digit numbers beginning with 139 and ending with 00.

```
import re

text = "139123456700"
pattern = r"^139d800$"

match = re.match(pattern, text)
if match:
    print("Success")
else:
    print("Fail")
```

Output:
Success
In the code, a text string "text" is first defined for testing regular expression matching. Then, a regular expression "pattern" is defined, where "^139" means starting with 139, "d8" means any eight digits, and "00$" means ending with 00. Then, the re.match() function is used for matching. If the match is successful, "Success" is outputted; otherwise, "Fail" is outputted.

1043

GPT-4

下面是你需要的python代码，这段代码中使用了python的re模块。

```
import re

def match_phone_number(string):
    pattern = "^139d600$"
    return re.match(pattern, string) is not None

# 测试
print(match_phone_number("13912345600")) # 输出: True
print(match_phone_number("15912345600")) # 输出: False
print(match_phone_number("13912345601")) # 输出: False
print(match_phone_number("139123456000")) # 输出: False
```

在这里，"^139d600$"是正则表达式。"^"表示匹配字符串的开始，"139"匹配字符串中的"139"，"d6"匹配任意六个数字，"00"匹配字符串中的"00"，"$"表示匹配字符串的结束。因此，这个正则表达式可以匹配到所有以139开头，以00结尾的11位数字。

Below is the python code you need. This code uses the python re module.

```
import re

def match_phone_number(string):
    pattern = "^139d600$"
    return re.match(pattern, string) is not None

# Test
print(match_phone_number("13912345600")) # Output: True
print(match_phone_number("15912345600")) # Output: False
print(match_phone_number("13912345601")) # Output: False
print(match_phone_number("139123456000")) # Output: False
```

In here, "^139d600$" is a regular expression. "^" represents the beginning of the string, "139" matches the "139" in the string, "d6" matches any six digits, "00" matches the "00" in the string, and "$" represents the end of the string. Therefore, this regular expression can match any 11-digit number that starts with 139 and ends with 00.

1134

Code

我用下面代码求二叉树的最大深度，但是答案不对，怎么回事呢?

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.left)
        queue = tmp
        res += 1
    return res
```

I use the following code to find the maximum depth of a binary tree, but the answer is incorrect. What could be the problem?

```python
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.left)
        queue = tmp
        res += 1
    return res
```

Responses

Model	Response	Elo
Qwen-7B-Chat (SFT)	你的代码是正确的，答案是正确的。在你的代码中，你首先将根节点添加到队列中，然后进入一个循环，直到队列为空。在每次循环中，你都会将队列中的所有节点的左子节点和右子节点添加到队列中。然后，你增加深度计数器res。最后，你返回深度计数器res。 Your code is correct, the answer is correct. In your code, you first add the root node to the queue, then enter a loop until the queue is empty. In each loop, you add the left child node and right child node of all nodes in the queue to the queue. Then, you increase the depth counter res. Finally, you return the depth counter res.	876

Qwen-14B-Chat (SFT)	你的代码会返回二叉树的节点数量，而不是最大深度。这是因为你在每次循环中都增加了res的值，而不是只在找到新的最深节点时增加。你可以通过在找到新的最深节点时才增加res的值来修复这个问题。以下是修复后的代码：
	`def maxDepth(self, root: TreeNode) -> int:`
	`  if not root: return 0`
	`  queue, res = [root], 0`
	`  while queue:`
	`  tmp = []`
	`  for node in queue:`
	`  if node.left: tmp.append(node.left)`
	`  if node.right: tmp.append(node.right)`
	`  queue = tmp`
	`  if len(queue) > res:`
	`  res = len(queue)`
	`  return res`
	Your code will return the number of nodes in the binary tree, not the maximum depth. This is because you increase the value of "res" in every loop, instead of only increasing it when a new deepest node is found. You can fix this problem by increasing the value of "res" only when a new deepest node is found. Here is the modified code:
	`def maxDepth(self, root: TreeNode) -> int:`
	`  if not root: return 0`
	`  queue, res = [root], 0`
	`  while queue:`
	`  tmp = []`
	`  for node in queue:`
	`  if node.left: tmp.append(node.left)`
	`  if node.right: tmp.append(node.right)`
	`  queue = tmp`
	`  if len(queue) > res:`
	`  res = len(queue)`
	`  return res`
		949

Qwen-14B-Chat (RLHF) 这个代码的问题在于它没有正确地处理左右子树的深度。具体来说，每次从队列中取出一个节点时，它会将该节点的左右子树都加入到队列中，然后将队列长度加一。但是，如果该节点的左右子树中有一个为空，那么这个节点的深度就应该是它所在队列的长度减一，而不是队列长度加一。 992

以下是修改后的代码：

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        res = max(res, len(queue))
    return res
```

在这个代码中，每次取出一个节点时，我们只将该节点的左右子树中非空的部分加入到队列中，并计算队列的长度。然后，我们将队列长度与当前最大深度进行比较，并更新最大深度。这样就可以正确地计算出二叉树的最大深度了。
The problem with this code is that it does not correctly handle the depth of the left and right subtrees. Specifically, when a node is taken out from the queue, it adds both its left and right subtrees to the queue and increases the length of the queue by one. However, if either the left or right subtree of the node is empty, then the depth of this node should be the length of the queue minus one, not the length of the queue plus one.

Here is the modified code:

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        res = max(res, len(queue))
    return res
```

In this code, when each node is extracted, we only add the non-empty parts of its left and right subtrees to the queue, and calculate the length of the queue. Then, we compare the length of the queue with the current maximum depth and update the maximum depth. This way, we can correctly calculate the maximum depth of the binary tree.

GPT-3.5

问题出在第11行代码，在将node.right添加到ttmp列表时，应该是tmp.append(node.right)而不是tmp.append(node.left)。修改后的代码如下：

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        res += 1
    return res
```

The problem lies in line 11 of the code. When adding node.right to the tmp list, it should be tmp.append(node.right) instead of tmp.append(node.left). The modified code is as follows:

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        res += 1
    return res
```

1090

GPT-4

你的代码中有一个错误：

你在这一行代码`if node.right: tmp.append(node.left)`中应该是想把右子节点加入到tmp列表中，但是你写的是node.left，这样就只会把左子节点加入到tmp列表中。正确的应该是node.right。

正确的代码如下：

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left: tmp.append(node.left)
            if node.right: tmp.append(node.right)
        queue = tmp
        res += 1
    return res
```

这种解法的思路是：每次迭代时，都把当前层的所有节点遍历一遍，同时把下一层的所有节点都加到下一次迭代的队列中。所以每次迭代完，深度res就加1。直到遍历的队列为空，此时的深度就是二叉树的最大深度。

1095

There is an error in your code:

In this line of code **if** node.right : tmp.append(node.left), you should be adding the right child node to the tmp list, but you wrote node.left instead, so only the left child node will be added to the tmp list. The correct code should be node.right .

The correct code is as follows:

```
def maxDepth(self, root: TreeNode) -> int:
    if not root: return 0
    queue, res = [root], 0
    while queue:
        tmp = []
        for node in queue:
            if node.left : tmp.append(node.left )
            if node.right : tmp.append(node.right )
        queue = tmp
        res += 1
    return res
```

The idea of this solution is: in each iteration, traverse all the nodes in the current level, and add all the nodes in the next level to the queue for the next iteration. So after each iteration, the depth res is increased by 1. Continue this process until the queue for traversal is empty, and the depth at that time is the maximum depth of the binary tree.

A.3 Analysis of Code Interpreter

Here we provide a case of comparison between Code LLaMA and Qwen-Chat. This case demonstrates the advantages of Qwen-Chat in processing tabular data and performing complex tasks.

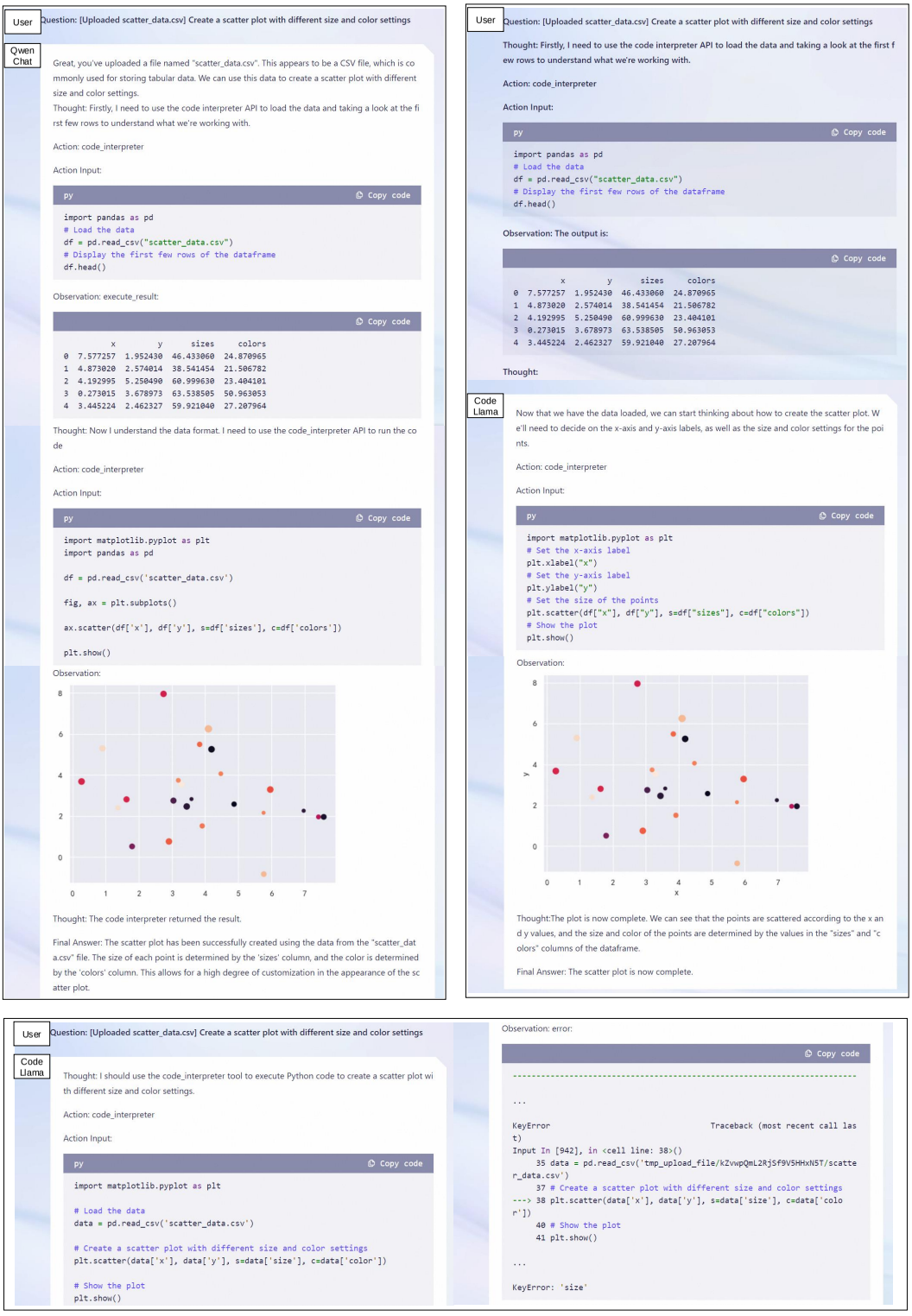

Figure 5: Example showcasing QWEN-CHAT's ability in using a code interpreter via ReAct prompting. The ReAct instruction is omitted for clarity. QWEN creates a two-step plan and first investigates the columns present in the CSV file before proceeding to draw the plot, as shown in the top-left figure. CODE LLAMA, however, attempts to draw the plot based on non-existent columns in its initial attempt, as seen in the bottom figure. CODE LLAMA can only reliably perform the task if the columns are provided in the user query, as shown in the top-right figure.