Second Place Overall Solution for Amazon KDD Cup 2024

Pengyue Jia*
jia.pengyue@my.cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

Jingtong Gao*
jt.g@my.cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

Xiaopeng Li*
xiaopli2-c@my.cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

Zixuan Wang*
zwang524-c@my.cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

Yiyao Jin*
yiyaojin2-c@my.cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

Xiangyu Zhao†
xianzhao@cityu.edu.hk
City University of Hong Kong
Hong Kong SAR, China

Abstract

This paper describes the solution from the AML_Lab@CityU team, who achieved second place in track 1 and track 5 (i.e., the overall track) and third place in track 2, track 3, and track 4 in Amazon KDD Cup 2024. We aim to construct a Large Language Model-based framework to answer diverse and complex online shopping questions with the multi-task and few-shot learning abilities of LLMs. This competition is challenging because of no training data provided, multi-task and complex online shopping questions, and sharp limitations on inference time and GPU memory. To tackle the above challenges, we introduce a pipeline containing three parts: base model selection, pre-trained model quantization, and prompt design. Our solution across all five tracks adheres to these three steps and demonstrates robust performance. It is worth noting that there is no fine-tuning in our solution, which broadens the usability of our pipeline. Our code is released online[1] for ease of reproduction.

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Recommender Systems, Large Language Models, Multi-Task Recommendations

ACM Reference Format:
Pengyue Jia, Jingtong Gao, Xiaopeng Li, Zixuan Wang, Yiyao Jin, and Xiangyu Zhao. 2018. Second Place Overall Solution for Amazon KDD Cup 2024. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

*Authors contributed equally to this research.
†Corresponding Author
[1]https://gitlab.aicrowd.com/gaojingtong/amazon-kdd-cup-2024-starter-kit

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Table 1: Statistics of ShopBench Dataset.

Tasks	Questions	Products	Product Category	Attributes	Reviews	Queries
57	20598	13300	400	1032	11200	4500

1 Amazon KDD CUP 2024

1.1 Dataset

To encourage the exploration of large language models' capabilities in multi-task online shopping scenarios, Amazon organizes the "Multi-Task Online Shopping Challenge for LLMs." In this competition, a new dataset named "ShopBench" has been introduced. ShopBench includes 57 online shopping tasks, with over 20,000 samples collected from the real-world Amazon shopping platform to evaluate the effectiveness of the participating teams' solutions. It is worth noting that all the data is in textual format, and every question has been converted into a text-to-text generation problem. The statistics of ShopBench are listed in Table 1.

However, the majority of the BenchShop dataset is not released to the participants and is only used for evaluation. The competition provides a development dataset with a total of 100 samples to help participating teams understand the problem format. In the development set, each sample includes the following fields:

- **input_field:** This field contains the questions.
- **output_field:** This field gives the ground truth answer.
- **task_type:** This field demonstrates the task type for the sample, e.g., "Retrieval".
- **task_name:** This field provides the hashed task name, e.g., "task0".
- **metric:** This field specifies the metric of this sample.
- **track:** This field specifies the track of the sample, e.g., "Shopping Concept Understanding".

For the test set, the samples only contain the "input_field field" in the development set with a field named "is_multiple_choice" specifies whether the current question is a multiple-choice question.

1.2 Tasks

There are five main types of tasks in ShopBench:

- **Multiple Choice:** Questions contain multiple options. The model needs to choose only one option.
- **Retrieval:** Given a list of candidate items, the model needs to retrieve all items that satisfy the requirement in the question.
- **Ranking:** Given a list of candidate items, the model needs to re-rank all items according to the question.

Table 2: Performance of Llama with different model size on the development dataset.

Model	Dtype	Temperature	Top_p	Accuracy
Llama3-8B-Instruct	bfloat16	0.0	0.9	0.4824
Llama3-70B-Instruct	AWQ quantization	0.0	0.9	0.7210

Table 3: Performance comparison between Llama3-70B-awq and Qwen2-72B-awq.

Model	Track1	Track2	Track3	Track5
Llama3-70b-awq	0.8013	0.7067	0.7064	0.7545
Qwen2-72b-awq	**0.8166**	**0.7141**	**0.7181**	**0.7685**

- **Named Entity Recognition:** Given a piece of text and an entity type, the model needs to extract phrases from the text as required.
- **Generation:** Given an instruction and a question, the model needs to generate texts to answer the question.

1.3 Execution Environment

Amazon KDD CUP 2024 provides the hardware environment for running the participants' solutions. Specifically, in the second phase, the competition offers 4 NVIDIA T4 GPUs for model inference, with the maximum model file size capped at 200GB. In addition, the competition imposes time limits on model inference for each track (70 minutes, 20 minutes, 30 minutes, 20 minutes, and 140 minutes for tracks 1-5, respectively). This is a stringent limitation, which, based on our experience, roughly requires an average response time of 0.6 seconds per question. Therefore, participating teams need to consider the hardware requirements and operational efficiency of their models in addition to performance.

2 Base Model Selection

2.1 Larger-Scale LLMs Deployment

In Phase 2, Amazon provides four T4 GPUs to process participating teams' solutions. Under such resource constraints, we can deploy either a small parameter model or a quantized version of a large parameter model. Here, we conduct experiments using Llama3-70b-awq and Llama3-8b to illustrate the performance of these two approaches. The corresponding results are presented in Table 2. The results indicate that quantized versions of bigger models can significantly enhance model performance compared to simply using a small base model. It is also worth noting that we used the vLLM [5] to deploy LLM and accelerate model inference.

2.2 Model Selection

We test various base models on the development dataset, including Llama [8], Gemma [7], Mistral [4], ChatGLM [10], Llama3 [1], Qwen [2], and Qwen2 [9]. The experiments show that Llama3 and Qwen2 perform better. Therefore, we continue to compare the performance of Llama3 and Qwen2 on the test set, which is shown in Table 3. We can find that, with the same prompts, Qwen2 is superior to Llama3. So, we take Qwen2 as the backbone model for further exploration.

Table 4: Hyperparameter tuning on Qwen2-72b-awq.

Max Data Length	Block_Size	N_Samples	Dataset Seed	Group Size	Accuracy
512	128	128	42	128	0.60864
128	128	128	42	128	0.61321
256	128	128	42	128	0.59657
128	256	128	42	128	**0.67484**
256	256	128	42	128	0.62959
128	512	128	42	128	0.64282
128	256	256	42	128	0.63509

3 Pre-trained Model Quantization

In this section, we will introduce the pre-trained model quantization process in the pipeline. We mainly focus on the hyperparameter tuning of LLM quantization. Our quantization implementation is based on Activation-aware Weight Quantization (AWQ) [6] following current work [3], which is implemented on AutoAWQ [2].

3.1 Hyperparameter Tuning

In the quantization process, the impact of hyperparameters on performance is quite significant. Specifically, the following parameters can be tuned:

- **Max Data Length:** This parameter controls the maximum length of one sample to be considered. Text exceeding this length will be truncated.
- **Block Size:** Controls the granularity of dividing each block. Typically, values of 128, 256, or 512 can be selected.
- **n_samples:** Determines the number of samples involved in the quantization process.
- **Dataset Seed:** Controls the random seed for the quantization.
- **Group Size:** Weight metrics will be divided into groups, with each group containing group_size columns.

We tune the hyperparameters on the given development set (100 samples). The results are shown in Table 4. We can find that different combinations of hyperparameters have significantly different performance on the development dataset, with the best and worst combinations differing by approximately 13% in accuracy.

We select the best-performing set of parameters for the test set experiments. The experimental results are listed in Table 5. We can see that our self-quantized model performs better on both track 1 and track 5. Additionally, it is worth noting that we also attempted to create more data to use as the quantization dataset, but this did not improve performance on the development dataset.

4 Prompt Design

In this section, we elaborate on methods to enhance LLM performance across various tasks, focusing on prompt design, hyperparameter tuning for LLM requests, and output formatting. We first present our overall inference structure, followed by detailed discussions on prompt design, hyper-parameter tuning, and output formatting techniques. Some related experimental results can be found in Table 6.

[2]https://github.com/casper-hansen/AutoAWQ

Table 5: Comparison experiments between Qwen2-72b quantized by Alibaba (Qwen2-72b-awq) and Qwen2-72b quantized by us (Qwen2-72b-awq-s) on 4 tracks.

Track	Model	Generation	Multi-Choice	NER	Ranking	Retrieval	Overall
1	Qwen2-72b-awq	0.6914	0.8456	0.6764	-	0.8296	0.7970
	Qwen2-72b-awq-s	0.6969	0.8453	0.7096	-	0.8226	0.7985
2	Qwen2-72b-awq	-	0.7619	-	-	0.6165	0.7438
	Qwen2-72b-awq-s	-	0.7540	-	-	0.5977	0.7345
3	Qwen2-72b-awq	0.7259	0.8632	0.8071	-	0.8246	0.8199
	Qwen2-72b-awq-s	0.7236	0.8617	0.8051	-	0.8259	0.8186
5	Qwen2-72b-awq	0.6465	0.7953	0.7449	0.8372	0.8034	0.7657
	Qwen2-72b-awq-s	0.6508	0.7976	0.7474	0.8334	0.8076	0.7685

Table 6: Experimental results for Prompt Design. "Qwen2-72b-awq" and "Qwen2-72b-awq-s" are the Qwen2 model quantized by Alibaba and by us. "Instruction(Tips)" describes whether the prompt includes text about tips. "Sub-task Division" represents the way to divide sub-tasks ("No" means without dividing, "Multi-Choices" means dividing all questions into Multi-Choices and Not-Multi-Choices, "All" means dividing all questions into multi-choice, NER, ranking, retrieval, and generation five categories). "Exemplar" demonstrates what types of examples are used in the prompt. "Param Tuning" represents whether we tune the parameters mentioned in Section 4.4.

Track	Model	Instruction (Tips)	Sub-task Division	Exemplar	Param Tuning	Generation	Multi-Choice	NER	Ranking	Retreival	Overall
2	Qwen2-72b-awq	No	No	Fixed	No	-	0.7486	-	-	0.4624	0.7128
2	Qwen2-72b-awq	Yes	No	Fixed	No	-	0.7402	-	-	0.5313	0.7141
5	Qwen2-72b-awq	Yes	No	Fixed	No	0.5094	0.7561	0.1236	0.7158	0.6468	0.6763
5	Qwen2-72b-awq	Yes	Multi-Choices	Fixed	No	0.6544	0.7954	0.7430	0.8339	0.8083	0.7679
5	Qwen2-72b-awq-s	Yes	Multi-Choices	Fixed	No	0.6521	0.7977	0.7474	0.8235	0.8045	0.7680
5	Qwen2-72b-awq-s	Yes	All	Fixed	No	0.6509	0.7977	0.7474	0.8334	0.8077	0.7685
5	Qwen2-72b-awq-s	Yes	All	Fixed+Dynamic	No	0.6509	0.8131	0.7474	0.8334	0.8066	0.7776
5	Qwen2-72b-awq-s	Yes	All	Fixed+Dynamic	Yes	0.6627	0.8132	0.8272	0.8360	0.8054	0.7815

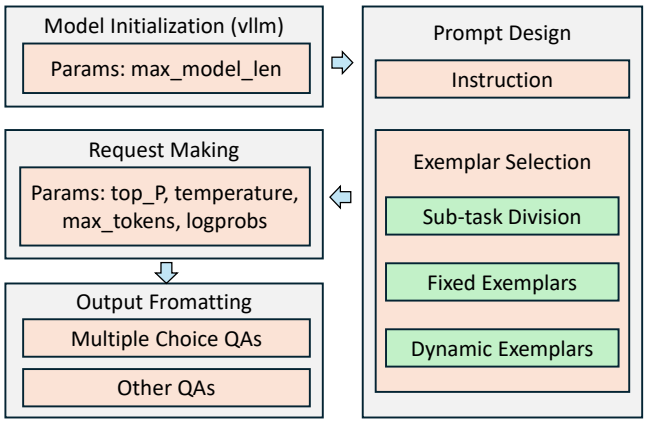

Figure 1: The overall inference structure.

4.1 Overall Inference Structure

As illustrated in Figure 1, the inference phase consists of four main stages: model initialization, prompt design, request making, and output formatting. The model initialization and request making stages primarily utilize functions from the vLLM for LLM initialization and reasoning, emphasizing hyper-parameter tuning. Conversely, the prompt design phase focuses on identifying various tasks and crafting prompts for performance enhancement, while the output formatting phase aims to refine the LLM's output to improve response effectiveness.

An example of the instruction

You are a very intelligent and helpful online shopping assistant for Amazon who can give reasonable answers or outputs to the online shopping questions. These are your Q&A history, please continue to answer questions concisely like these until the answer or output to the last question is finished. If your answer is excellent, you will get tips.

Figure 2: An example of the instruction.

4.2 Instruction Design

We find that few-shot in-context learning is highly effective under constraints of limited reasoning time and training samples, especially when our computing resources are scarce. Thus, our prompts primarily consist of instructions detailing task requirements and relevant exemplars. An example instruction is shown in Figure 2. Typically, the instruction content directly explains task and output requirements without a fixed format. The core concept here is to provide the LLM with "tips" within the instruction, as suggested by some people on the internet [3]. Experiments demonstrate that these tips consistently enhance performance across various tracks, though the degree of improvement varies.

[3]https://minimaxir.com/2024/02/chatgpt-tips-analysis/

Table 7: Experimental results for dynamic exemplars

Model	Generation	Multi-Choice	NER	Ranking	Retrieval	Overall
Base	0.6508	0.7976	0.7474	0.8334	0.8076	0.7685
Dynamic_1	-	-	-	-	-	Timeout
Dynamic_3	0.6508	0.8098	0.7474	0.8334	0.8076	0.7756
Dynamic_4	0.6508	0.8131	0.7474	0.8334	0.8076	0.7776

4.3 Exemplar Selection

Exemplars play a critical role in few-shot learning. This section outlines our strategy for exemplar selection.

4.3.1 Sub-task Division. Selecting appropriate samples for different tasks is crucial in few-shot learning. A straightforward approach involves dividing tasks into sub-tasks (e.g., generation, multiple selection, retrieval) and providing subtask-specific exemplars during inference. We identified fixed patterns (i.e., commonly used words or strings) in the provided training samples to classify different subtasks and designed exemplars accordingly.

4.3.2 Fixed Exemplars. For each subtask, we selected exemplars from the related category in the training set. However, not all subtasks perform best with category-specific exemplars. Experiments revealed that using diverse exemplars from different subtasks can yield better results, particularly for tasks involving retrieval and generation questions. This may be because the diversity of exemplars helps maintain output diversity in tasks that generate new content.

4.3.3 Dynamic Exemplars. Our approach incorporates dynamic exemplars into prompt construction. Traditionally, examples within prompts are fixed and unchangeable, and our goal is to address this limitation. During the shopBench benchmark, batches of questions exhibit unique distributions due to varying problem types. To address this, we propose a dynamic sampler that incorporates different examples in our prompts to enhance performance.

For each batch request, we store the first instance, including its problem formulation and the model-generated answer. In subsequent batch requests, we append this example to the end of the prompt, completing the dynamic examples for each batch request. We implemented this strategy in tracks 1, 4, and 5 for multiple-choice questions and verified its effectiveness.

Table 7 shows the experimental results for dynamic exemplars on track 5 to give a comprehensive analysis. It is worth noting that we only use this technique on multiple-choice questions due to time limitations. There are four variants in this experiment:

- **Base:** This variant does not contain dynamic exemplars.
- **Dynamic_1:** This variant uses the dynamic exemplars technique, and the dynamic exemplar is updated each batch.
- **Dynamic_3:** Same as Dynamic_1, except the dynamic exemplar updating frequency occurs every three batches.
- **Dynamic_4:** Same as Dynamic_1 except the dynamic exemplar updating frequency occurs every four batches.

From Table 7, we can draw the following conclusions: (1) From Dynamic_1, we know that updating the dynamic exemplar for every batch is very time-consuming and cannot complete the inference task within the allotted time in track 5. Therefore, we try to reduce the inference time by updating the dynamic exemplar at intervals of multiple batches. (2) By comparing Base and Dynamic_3, we find that using dynamic exemplars can effectively improve the accuracy of multiple-choice questions (from 0.7976 to 0.8098), demonstrating the effectiveness of dynamic exemplars. The reason for this improvement is that there is a distribution difference between the examples in the development set and the real test set questions, and the dynamic exemplar helps to mitigate this gap. (3) By comparing Dynamic_3 and Dynamic_4, we find that Dynamic_4 achieves better results. A possible reason for this improvement is the randomness in the arrangement of questions in the test set.

4.4 Hyper-parameters Tuning

We primarily tested the impact of different hyper-parameters (shown in Figure 1) on Track 5, as Track 5 comprehensively includes all types of questions and tasks. Specifically, we mainly fine-tune the following hyper-parameters:

- **Max Context Length (max_model_len in vLLM):** This parameter controls the model context length. Increasing this parameter from 4096 to 8192 can obtain better performance while increasing it from 8192 to 12288 cannot gain improvement continuously.
- **Max New Tokens (max_tokens in vLLM):** Controls the maximum number of tokens to generate per output sequence. For Multiple-Choice Questions (MCQ), NER, ranking, and retrieval questions, large values (e.g., 50) can lead to redundancy outputs. So, in track 5, we set the max_tokens in MCQ to be 1, NER and ranking to 15, retrieval to 10, and generation to 65.
- **Temperature:** Controls the randomness of the sampling. To ensure the reproducibility, we set it to 0.
- **top_p:** Controls the cumulative probability of the top tokens to consider. We set it to 0.9.

4.5 Output Formatting

To ensure the output is more standardized and accurate, we apply the following rules to filter and modify the answers:

- For multiple-choice questions, we verify if the generated token is a valid option. If not, a default option will be chosen.
- For non-multiple-choice questions, we check if the generated text contains any unnecessary information, such as the word "question" at the beginning or trailing spaces at the end.

5 Efficiency

In this section, we discuss the efficiency problem we encountered during this competition, which mainly includes the following four parts: Reasoning time for different models, the impact of the length of max token in generation tasks, the influence of prompt length on efficiency, and the influence of max model length to efficiency.

5.1 Efficiency of models

In Figure 3 (a), we present the time consumption for each track within our experiment. During the experiment, we selected different quantized versions of backbone LLMs such as Llama3 and Qwen2. We observed that due to the varying question distributions across tracks, the inference times also varied. Notably, in track 4, we encountered a severe time limit issue. Specifically, we faced a dilemma: while Llama3's reasoning speed was nearly 10% faster than Qwen2's,

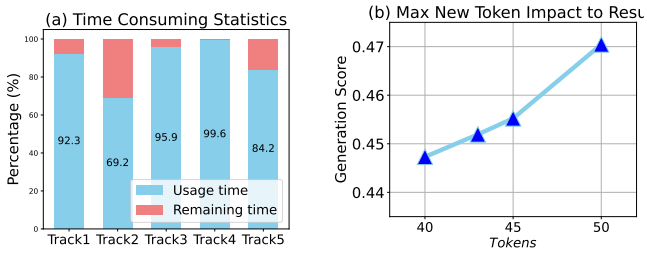

Figure 3: (a) The ratio of time we spend on each track to the total time limit. (b) Generation task scores corresponding to different max tokens in track 4.

Qwen2 generated better results with the same prompts. We could use longer prompts to improve in-context learning ability, but this was not feasible given the time constraints. Consequently, we designed two approaches: the first utilized Llama's method with more shot prompts, and the second employed Qwen2's method with fewer shot prompts. After extensive experimentation and evaluation, we chose Qwen2's method, achieving a total score of 0.715 in track 4, which placed third in the leaderboard.

5.2 Max Token Influence in Generation Task

Except for Track 2, the remaining four tracks generated questions based on the provided examples, including tasks such as title generation, keyphrase extraction, product explanation, and translation. A key finding of this study is that a larger number of tokens in the generation task correlates with a higher score. The results, illustrated in Figure 3 (b), indicate that setting the maximum token count to 50 in Track 4 yields the highest scores. This correlation is logical as a higher token count encompasses more useful information, thereby enhancing metrics such as "sent-transformer" and "bleu". However, a higher token count can lead to timeout issues. Therefore, within the given time limit, we experimented with different settings for different tracks: 40 for Track 1, and 100 for Track 2, 50 for Track 4, and 65 tokens for Track 5.

5.3 Prompt Length Influence to efficiency

As shown in Section 4.3, we include a few-shot examples strategy within our prompts to guide the model toward generating more reasonable outputs. Our key finding is that selecting a higher number of relevant examples consistently enhances performance across tracks. However, due to time constraints, we must carefully choose the number of examples included in our prompts. Taking Track 4 as an example, we use different few-shot prompts for various problem types. In our results, we selected 4 examples for the multiple-choice prompt, 3 examples for the ranking problem prompt, and 6 examples for the generation prompts. This decision balances the number of examples with the "max new tokens" parameter, as previously discussed. Although more examples could further enhance results, we are constrained by the significant time limitations in this track and therefore cannot add more.

6 Discussion

There are a few takeaway notes in our solution: (1) Under the same hardware conditions, the quantized version of the larger model (Llama3-70b-awq) performs significantly better than the smaller base model (Llama3-8b). (2) The choice of quantization parameters affects model performance, but the extent of this impact requires further experimentation. (3) Prompts have a substantial impact on model performance. Using targeted prompt templates for specific tasks and employing tips to motivate the model can yield consistent benefits. (4) Dynamic exemplars can effectively mitigate the issue of distribution discrepancies between the validation set and the test set, thereby improving the model's accuracy in answering questions. (5) Output formatting can effectively standardize the output and improve the QA performance.

7 Conclusion

Amazon KDD CUP 2024 focuses on leveraging the capabilities of large language models to assist multi-task online shopping services. In this paper, we present the solutions that achieve second place in tracks 1 and 5 and third place in tracks 2, 3, and 4. Our approach comprises three components: base model selection, pre-trained model quantization, and prompt design. We select Qwen2-72b as our base model and employ AWQ quantization to ensure the model meets hardware and inference efficiency constraints. In addition, we use techniques such as instruction design, exemplar selection, hyper-parameter tuning, and output formatting to achieve leading performance.

References

[1] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[3] Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. How good are low-bit quantized llama3 models? an empirical study. *arXiv preprint arXiv:2404.14047* (2024).

[4] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[5] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[6] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. In *MLSys*.

[7] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295* (2024).

[8] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[9] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671* (2024).

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