HUIXIANGDOU: OVERCOMING GROUP CHAT SCENARIOS WITH LLM-BASED TECHNICAL ASSISTANCE

Huanjun Kong **Songyang Zhang** **Kai Chen**

Shanghai AI Laboratory

ABSTRACT

In this work, we present HuixiangDou[1], a technical assistant powered by Large Language Models (LLM). This system is designed to assist algorithm developers by providing insightful responses to questions related to open-source algorithm projects, such as computer vision and deep learning projects from OpenMMLab. We further explore the integration of this assistant into the group chats of instant messaging (IM) tools such as WeChat and Lark. Through several iterative improvements and trials, we have developed a sophisticated technical chat assistant capable of effectively answering users' technical questions without causing message flooding. This paper's contributions include: 1) Designing an algorithm pipeline specifically for group chat scenarios; 2) Verifying the reliable performance of text2vec in task rejection; 3) Identifying three critical requirements for LLMs in technical-assistant-like products, namely scoring ability, In-Context Learning (ICL), and Long Context. We have made the software and source code available at github.com/InternLM/HuixiangDou to aid in future research and application. HuixiangDou is applicable to any group chat within IM tools.

1 INTRODUCTION

Authors of open-source projects often set up user groups on IM tools(like WeChat, Slack, Discord, etc.) for discussing project-related technical questions. As the number of users gradually increases, the maintainers, aiming to reduce the time spent on answering user questions while ensuring these questions are addressed, tend to pin some content or set up a bot to automatically answer FAQs. However, user inquiries are strongly correlated with their local development environments, and most messages in the group are unrelated to the project. However, traditional NLP solutions can neither parse the users' intent nor often provide the answers they desire.

ChatGPT, a large language model service from OpenAI, has a good performance in multiple test sets and natural language communication. However, directly integrating ChatGPT into group chats could lead to more severe issues.

- ChatGPT is designed for single-user chat. If it responds too many messages within a group, it may impact others' experience and cause them to leave the group.
- For really valuable queries such as code implementation principles and modification methods, ChatGPT fails to provide correct answers. This is because its training data comes from the public internet, not domain-specific knowledge, and its data cannot be updated immediately with code modifications.
- Even though ChatGPT exhibits high accuracy in numerous datasets, it still faces the issue of hallucination. For example, asking "Who is the author of ncnn?" can yield an incorrect response related to "Nvidia Compute Library".

Hence, a technical assistant operating in group chats has different requirements.

[1]HuixiangDou is a dish from the Chinese classical story 'Kong Yiji'. It's a meal Kong Yiji would routinely order from a local tavern, reflecting his humble conditions yet exalted spirit.

Target true help-seekers Technical assistant should not respond to non-technical content, such as politics, chit-chat, or personal information. They are only activated to provide responses to technical inquiries when users genuinely require assistance.

Strictly no hallucination Even a single instance of hallucination could make users perceive the bot as unreliable from a product perspective. Therefore, the system is implemented to avoid creating any false impressions of understanding.

Understand domain-specific knowledge Possessing exclusivity not found in the public internet is the fundamental value of an assistant. Simultaneously, it can update the version content of the knowledge base at a relatively low cost.

No rush for response Users might ask questions late at night without having much expectation for response time. Therefore, we can adopt more complex processing procedures.

In addressing these unique needs, we started with a basic version and arrived at our current solution after making two improvements. For easy understanding, we will metaphorically name these improvements after cold weapons.

1. Baseline - ”Dagger”. Directly fine-tuning the LLM to handle the issues.
2. Improved version - ”Spear”. First, find the problem’s key point, then tackle it with LLM. As the strategy is like attaching a dagger to a wooden pole to eliminate distractions and locate the problem.
3. Final version - ”Rake”. Find multiple target points, bundle them, and then process with LLM. Compared to the spear, the tines of a rake cover a larger target area.

It’s worth noting that the entire roadmap has evolved step by step, and the earlier versions all have their shortcomings.

2 Evolution of Approach

2.1 Baseline

Upon receiving a group message, the baseline version processes it and hands it over to the fine-tuned LLM for a response, which is then returned to the group chat. Unfortunately, there were significant issues with hallucinations in baseline.

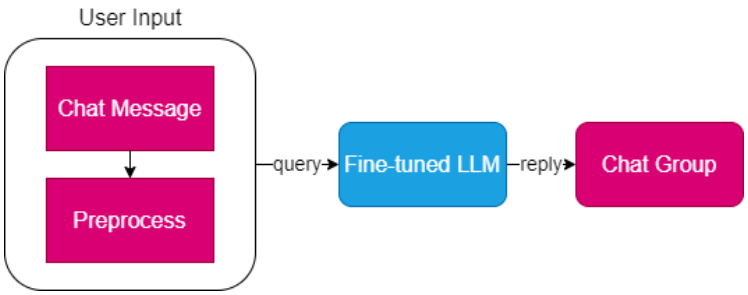

Figure 1: Baseline directly process user question with fine-tuned LLM.

2.1.1 Chat Message Preprocess

In a chat group, multiple users may pose questions and communicate among themselves. However, the typical LLM chat template generally only accommodates three roles: system, user and bot. Hence, we simplify the handling of messages within the group by categorizing them based on user ID. This approach renders the concept of multiple users imperceptible to the LLM.

Given that users are unlikely to describe their problem completely in one go, we pack multiple consecutive messages into a single one. In the process, we disregard interstitial elements such as images, emojis, and voice messages.

2.2 Improved Version

The purpose of this approach is to eliminate hallucinations. In this scenario, hallucinations come from two parts: user gossip and the model itself (where the model's training data and domain knowledge are not aligned).

Obviously, models with robust In-Context Learning capabilities can assuredly mitigate internal hallucinations through search mechanisms. Thus, building upon the baseline, we added two pipelines.

- Two-staged Reject Pipeline is used to dismiss casual chat-like discourse.
- Response Pipeline, used for finding answers to real problems, consists of Retrieval-Augmented Generation (RAG) and LLM prompt techniques.

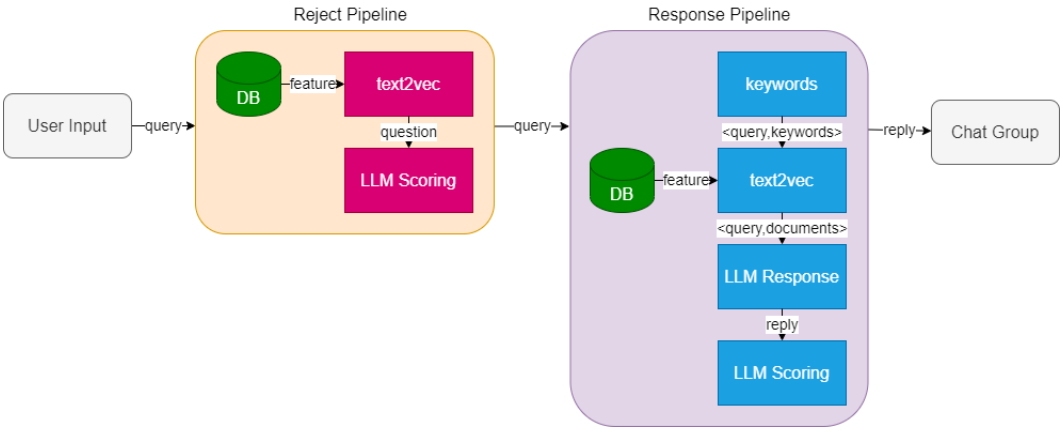

Figure 2: Improved version aims at diminish model hallucination.

2.2.1 Reject Pipeline

Refusal to Answer Based on Text2Vec LangChain Contributors (2023a) and wenda wenda Contributors (2023) were originally used for RAG. After repeated tests, we think their retrieval abilities are normal, but surprisingly suitable for telling whether the question deserves to be answered. Undesired question are topics that are too distant from the knowledge base.

Refusal to Answer Based on LLM Scoring Because the text2vec model judges topic similarity, it's easy to be influenced by tone words in the chat group question. From the perspective of the text2vec model, there is a high degree of similarity between "This development board is very good" and "This board is poorly designed". Influenced by moral and other factors, humans do not believe that these two sentences express the same meaning.

2.2.2 Response Pipeline

Extract Keywords User queries often contain many modal words, which can greatly impact the precision of text2vec models. Therefore, we cannot directly use original queries for text2vec search. As LLM excels at NLP part-of-speech segmentation tasks, we leverage it to directly extract keywords and phrases from the query.

Search Document Snippets We mixedly use LangChain and wenda to retrieve domain-specific knowledge. In this scenario, our search result is a list of document snippets. To fully utilize the context length of the model, we also employ LLM scoring to judge the relevance between the query

and the document. This helps avoid any distractions for the LLM from irrelevant inputs. It's evident that LLM's In-Context Learning capability is extremely crucial for this scenario.

Due to the varying performance of different text2vec models, the Response Pipeline does not share a feature database with the Reject Pipeline.

LLM Response LLM responds to queries based on background knowledge. Ultimately, we use LLM scoring to evaluate the relevance between the response and the query. If the relevance is low, assistant will not respond.

2.3 Final Version

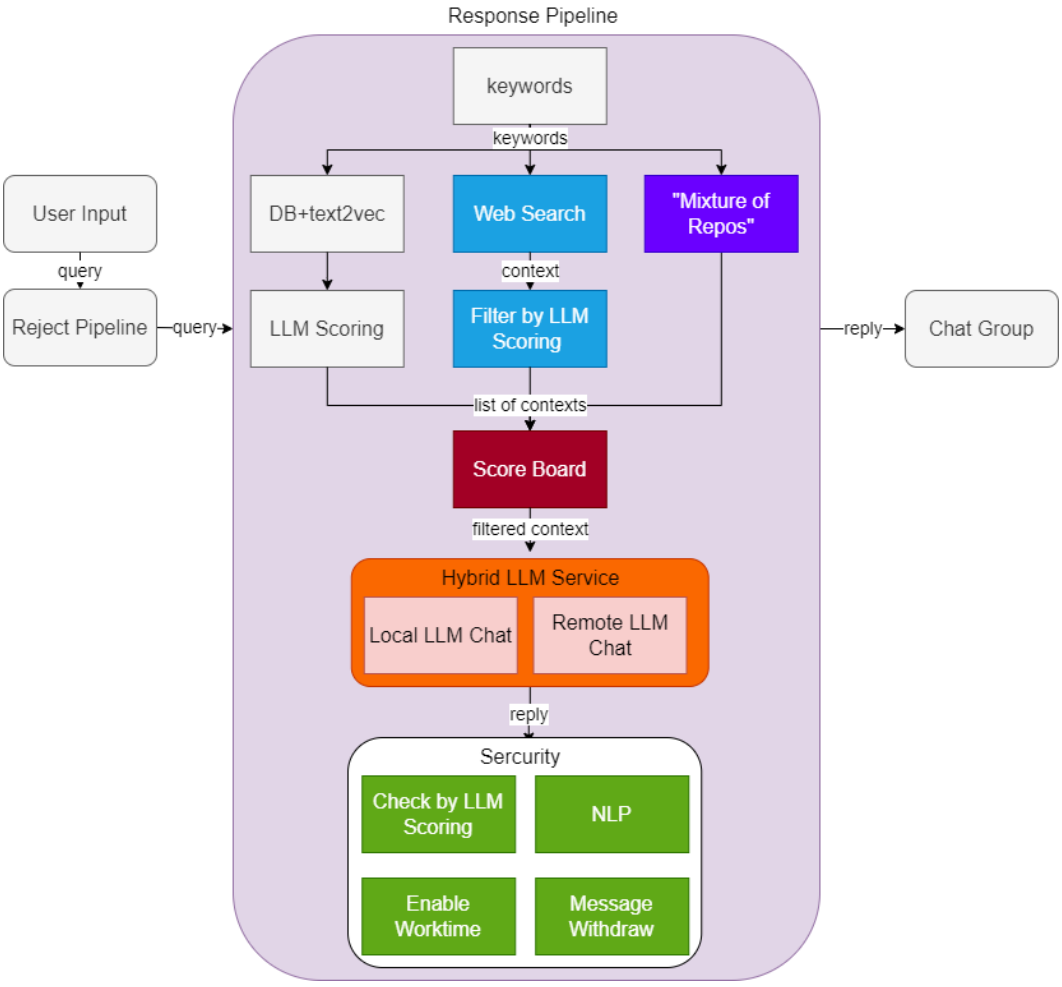

Figure 3: Rake approach would capture multiple available context.

Improved Version guarantees a high precision in refuse-to-answer scenarios, preventing message flooding that could lead to user dissatisfaction. However, in actual operation, due to the mediocre performance of text2vec, the technical assistant rarely answers questions.

Final Version strengthens long context ability of the chat model, simultaneously extending the Response Pipeline in three aspects to increase the likelihood of providing good answers. It maintains a high precision in refuse-to-answer scenarios while improving the precision of the assistant through LLM prompts and search enhancement.

Web search We first retrieve multiple search results (depends on the maximum token length supported by the model) and then utilize LLM scoring to filter the results associated with the question. These results are finally packed into the background document. This will return illegal content, so safety filtering is also necessary.

"Mixture of Repos" Search engines face the entire spectrum of internet information, but the background information implicit in the group chat technical assistant isn't fully utilized. For instance, users wouldn't actually ask mmdetection Chen et al. (2019) questions in the opencompass Contributors (2023c) user group.

Based on sourcegraph McColl et al. (2013), we built a unique search engine for each repository, routing queries from different groups accordingly. This improvement enables the assistant to answer difficult questions that internet searches can't locate, which we discuss further in our LLM paging experiments.

In order to avoid misunderstanding, the original intention of MoE Gale et al. (2022) is to extend parameters, not specifically to handle domain knowledge.

Score Board The long context capability of the model is valuable. We can't feed all documents into the LLM at one time. This step involves deciding which pieces of information will make up the final input for the LLM Chat based on prior knowledge. For instance, for PyTorch-related questions, we tend to look up the official PyTorch documentation rather than some tech blog.

Hybrid LLM Service Our product focuses more on cost-efficiency and does not insist on a single model possessing all capabilities. We treat LLM chat as an RPC (Remote Procedure Call) service that can internally integrate multiple models for use as needed. This step is optional. HuixiangDou adopts this approach to enhance the capability of long context.

Security Many regions emphasize the security of AI applications. To ensure foolproof safety, we implemented four seat belts:

- Check all string variables and their association with prohibited topics based on LLM scoring to prevent the generation of illegal content.
- Integrate traditional security services to check whether the assistant's responses are illegal.
- Set working hours for the assistant to ensure all activities are under human supervision.
- Everyone can withdraw HuixiangDou's response if they deem inappropriate.

3 Experiments

3.1 Baseline Fine-tuned Model

3.1.1 Base Model Selection

Due to resource limitations, we cannot train from scratch and must select a base model for fine-tuning. Our selection criteria are as follows.

- Understanding domain-specific terminologies. That means training data includes the vocabulary needed for business operations. Otherwise, we believe that the required results cannot be calculated using attention score.
- Long context. Since we can use ReRoPE Su (2023) or dynamic NTK emozilla and bloc97 (2023) for expansion, a model supporting RoPE can be considered capable of handling long context.
- In-Context Learning (ICL) and stable scoring ability.

3.1.2 Data Preparation

Our training data comprises 28,000 QA pairs, which are made up of three parts:

1. We cleaned some existing group chat data, removed personal information, and divided the dialogues into the QA format required for training. Among them, there are about 8,000 question-answer pairs.

2. For unanswered questions, we constructed responses using a larger LLM. These account for approximately 12,000 of the total.

3. We also scraped closed issues from GitHub, amounting to about 8,000 entries.

3.1.3 Train and Test

We used the XTuner Contributors (2023d) qLoRA method to fine-tune on the 7B and 13B models respectively. Our learning rate is 2e-5 and epoch is 5. Regardless of the combination, there were significant issues with hallucinations. In the best version, the model learned colloquial expressions from users in WeChat groups, rather than technical answers.

Fine-tuned LLM Chat Examples

User:
Can mmdet3.0 use version 0.7 of mmcv ?
Assistant:
mmdetection3.0 and mmcv version 0.7 are not compatible due to differences in certain features. Mmdetection 3.x requires mmcv 2.0 or a higher version. If you need to use an older version of mmcv, you may consider downgrading to mmdetection 2.x.

Hallucination Example:
User: Does mmdet3.0 support mmcv0.7 ?
Assistant: There's no need for that, the version of mmcv should be between 1.3.3 and 1.5.0.

Response Style Example:
User: mmyolo uses the coco format, right ?
Assistant: Haha, your understanding is very precise. Yes, mmyolo indeed utilizes the coco format.

Figure 4: Fine-tuned LLM examples.

We believe that the biggest issue lies in data quality. Other users answer the questions casually and unprofessionally. In terms of domain-specific questions, the answers provided by LLM are not correct.

3.2 RAG in Reject Pipeline

We manually annotated hundreds of user contents, with human judgement determining whether they were related to domain-specific knowledge. We then used different text2vec models to construct a database and test the accuracy of refusal to answer.

Model	Precision	Recall
text2vec-large-chinese	0.99	0.92
text2vec-bge-large-chinese	0.95	0.81

Table 1: Refusal to answer task on different text2vec models.

We also examined the impact of various text split methods on precision, including `langchain.MarkdownHeaderTextSplitter`, `langchain.CharacterTextSplitter` and their combined implementation. Experiments showed that the impact of the split method on the precision of refusal to answer can be disregarded.

3.3 LLM SCORING

LLM Scoring has been utilized in intent determination, the evaluation of relevance between questions and background, as well as within security contexts.

This is implemented by determining the final score of the task. In engineering practice, we often use integers and booleans instead of strings to determine the result of an if statement. Below is the LLM scoring prompt.

System prompt used in Scoring

Prompt:
Determine whether the following sentences are topical interrogative sentences, with results ranging from 0 to 10. Provide scores directly without explanation.

```
Scoring standards:
A score of 10 for sentences with subject, predicate, and object that are
interrogative;
points deducted for missing subject, verb, or object; a score of 0 for declarative
sentences;
a score of 0 for non-interrogative sentences.
```

New question "{}", what is the score? Provide scores directly without explanation.

We randomly selected 1,303 domain-related queries and use InternLM2-7B to estimate the likelihood of the query being a question—the higher the score, the more likely it is a question. The results showed that 11.6% of the content were user questions, which is in line with common sense.

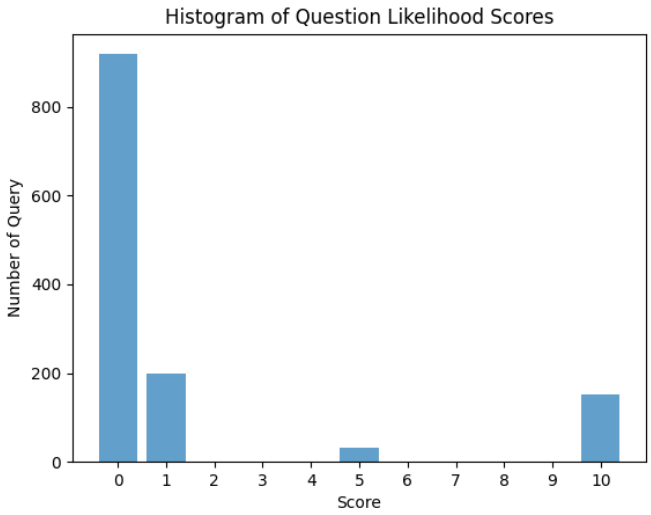

Figure 5: Question likelihood with InternLM2-7B.

While elaborating on the problem can enhance the model's performance, the model's perspective of the world doesn't straightforwardly equate to that of a human. For instance, a submarine can "swim," but this is not equivalent to human swimming.

We extracted 11,362 sentences from the content sent by users. To improve the precision of Refusal to Answer, we included scoring examples in the prompt. However, after adding these examples, the scores for all 7,753 pieces of data increased.

But in reality, more than 80% of group chat consists of idle chatter. If precision improves, the scores should present a polarized state of minor increases and major decreases.

Scoring examples

Part of Prompt:

```
Here are some examples:
Question "Excuse me, how should mmdeploy be installed?", Score: 9
Question "How to apply for modification of rpm?", Missing subject, Score: 7
Question "Please check if the environment is installed with your revised version",
It's a declarative sentence, Score: 0
Question "If you treat bot as a living person, everything will go smoothly", It's
not a question, Score: 0
```

3.4 Long Context

Based on our experience in Chinese-English bilingual scenarios, the token length of a search result can exceed 11k. Considering the prompt and historical dialogue, the model's max token length should be more than 16k. Only with 32k can we achieve relatively good results.

Considering the prohibitive training cost of YaRN Peng et al. (2023), we optimized ReRoPE's inference performance using Triton Contributors (2019), also introducing dynamic quantization.

Model	Max Token Length	Precision	GPU Memory
baseline	4k	1.0	*65GB
baseline+ReRoPE	14k	1.0	79GB
baseline+ReRoPE+Triton+Quantization	40k	1.0	75GB

Table 2: Passkey test results of using different methods on openbuddy-llama2-13B-v8.1-fp16. To optimize speed, LMDeploy automatically pre-allocates memory based on the GPU, hence the base model occupy 65GB.

Eventually, we achieved support for 40k token length on an A100 80G graphics card. The pull request[2] has been merged into LMDeploy Contributors (2023b). Table 2 is our precision test report for passkey retrieval, with the base model being openbuddy-llama2-13B-v8.1-fp16.

3.5 LLM Paging

Suppose we want to make LLM understand an entire repository via prompts. Even the latest 192k context length can't accommodate the full source code of OpenCompass. During ReRoPE optimization, we also realized that the transformer kv cache and attention score mechanism severely limit the maximum context length.

Inspired by the operating system paging mechanism, we compressed the Python module into a single description, thereby shrinking the OpenCompass project within 120k. For user technical queries, we let LLM decide which modules to view, then extract the module source code for secondary inquiries. However, in practice, LLM only finds partial source code using a 128k context, and user questions may involve multiple knowledge points. Here is an LLM Paging example without any web search nor RAG results.

4 Other Attempts

4.1 NLP in RAG

Since the capabilities of the text2vec model are limited, we have tried to simplify the query and document with NLP methods. For example, inputting "How to install mmdet and mmcv" will identify CC part of speech, thereby decomposing into two simple questions. But in actual operation, we encountered more difficult problems.

[2]See https://github.com/InternLM/lmdeploy/pull/718

LLM Paging Example

User:
How do I output the results of triviaqa 5shot in a summarizer?
Real Solution:
Add an element: `'triviaqa_5shot'` to the array `summarizer['dataset_abbrs']` in the configuration file.
Assistant:
First, you need to ensure that the results of "triviaqa 5shot" are stored in your code in an appropriate data structure (such as a dictionary or list), and that these results can be accessed from your data processing or loading section.
Next, you need to find or create a method within the relevant summarizer class that handles and formats the "triviaqa 5shot" results. This might involve parsing, computing, or transforming data so they can be correctly included in the output of the summarizer.
Finally, you need to insert code into the output methods of the summarizer (like _form_table or _format_raw_txt) so that the "triviaqa 5shot" results are included when generating summaries. This might involve adding new columns or rows to a table or text, or modifying existing data to include new information.

Figure 6: LLM Paging only searched the source code of `opencompass.summarizer`, and was not aware of the existence of the config file.

1. Domain-specific part-of-speech tagging lacks precision. For example, in the field of deep learning, the part of speech for "deploy" depends on the context, which is different from daily communication.

2. Bilingual problems. HanLP He and Choi (2021) exhibits subpar performance in English, and other well-known projects do not support Chinese. Utilizing translation APIs to bridge this gap in bilingual models poses further complications. Due to the lack of appropriate translations for certain terms, it can result in significant misinterpretations, such as with the term "transformers".

4.2 Prompt Technics

To improve the precision of LLM Scoring and effectiveness of question responses, we attempted numerous techniques. Unfortunately, most of these did not work well in this scenario.

4.2.1 Rephrase and Respond

Rephrase and Respond Deng et al. (2023) attempts to enhance the prompt using LLM, but this is constrained by the understanding ability of the base model, making it incapable of extending this technique to interrogative sentences. Otherwise, it would lead to confusion in the LLM. Here is an scoring example.

Rephrase and Respond Example

User:
"Determine whether the following sentences are topical interrogative sentences, with results ranging from 0 to 10. Provide scores directly without explanation."
Rephrase and expand the question, and respond.
Assistant:
1

Figure 7: RaR prompt is not applicable to interrogative sentences.

4.2.2 REACT

ReAct Yao et al. (2023) utilizes training data to potentially generate fixed-format json results based on inputs, which are then employed to invoke tools such as search engines. However, from a product perspective, being unable to debug indicates a significant risk. In practical use, search behaviors are often triggered even for simple queries. Considering that this approach requires training data, we don't deem it cost-effective for practical use.

5 CONCLUSION AND LIMITS

In this work, we demonstrated the feasibility of using text2vec for refusal response, and multiple search methods can substantially mitigate the hallucination caused by LLMs.

We believe that as long as an LLM has the following capabilities, coupled with business functionality, it can sufficiently address most demands within group chat scenarios:

- Understanding domain-specific terminologies.
- Supporting a minimum token length of 16k.
- Scoring capability.
- In-Context Learning.

However, as users' questions become more professional, it's increasingly difficult to provide satisfactory responses based on the prompt and search method. This necessitates that the LLM truly understands the source code in the repository. We think efficient further pretrain is the next stage solution.

Due to the limitations of the ChatML OpenAI (2022) format, we have merely divided the group messages according to the user, which in fact has led to a significant loss of contextual information. The new chat format should fully expressing the context of the problem, the historical messages of the speaker, and the remarks.

Additionally, users are very fond of first sending log screenshots before asking questions. Many valuable contexts are contained within these images, but HuixiangDou does not support multimodalities. We've tried various open-source or commercial OCR methods, but the results are not ideal. We will work on it.

6 ACKNOWLEDGMENTS

- We would like to express our gratitude towards the OpenMMLab users and ncnn contributors for their understanding and tolerance of the numerous bugs in the technical assistant.
- We are grateful to the teams at OpenCompass, XTuner, and LMDeploy for their guidance during the exploratory phase of the project.
- Our thanks also go to Moonshot AI and Xinran Xu for providing a free 128k context LLM API.
- We extend our appreciation to Jianlin Su, the author of RoPE, for his profound insights into the structure of transformers.
- Finally, we want to thank libowen@pjlab.org.cn and liukuikun@pjlab.org.cn for their teachings and ideas on NLP and huwenxing@pjlab.org.cn for his method of integrating WeChat and Siyue Zhao for the proofreading on this Report.

REFERENCES

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

LangChain Contributors. Langchain: Building applications with llms through composability, 2023a. URL https://github.com/langchain-ai/langchain.

LMDeploy Contributors. Lmdeploy is a toolkit for compressing, deploying, and serving llms, 2023b. URL https://github.com/internlm/lmdeploy.

OpenCompass Contributors. Opencompass: A universal evaluation platform for foundation models, 2023c. URL https://github.com/open-compass/opencompass.

Triton Contributors. Development repository for the triton language and compiler, 2019. URL https://github.com/openai/triton.

XTuner Contributors. Xtuner: A toolkit for efficiently fine-tuning llm, 2023d. URL https://github.com/internlm/xtuner.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves, 2023.

emozilla and bloc97. Dynamically scaled rope further increases performance of long context llama with zero fine-tuning, June 2023. URL https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases/.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts, 2022.

Han He and Jinho D. Choi. The Stem Cell Hypothesis: Dilemma behind Multi-Task Learning with Transformer Encoders. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5555–5577, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.emnlp-main.451.

Rob McColl, David Ediger, Jason Poovey, Dan Campbell, and David Bader. A Brief Study of Open Source Graph Databases, 2013.

OpenAI. The official python library for the openai api, 2022. URL https://github.com/openai/openai-python/blob/e389823ba013a24b4c32ce38fa0bd87e6bccae94/chatml.md.

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

Jianlin Su. The upgrade path of transformer: 12, infinite extrapolation with rerope?, Aug 2023. URL https://spaces.ac.cn/archives/9708.

wenda Contributors. wenda: A large language model (llm) invocation platform, 2023. URL https://github.com/wenda-LLM/wenda.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.