**SC4052 <Cloud Computing> Project Report**

**Topic 4: Cloud Computation by Crowdsourcing: <PromptCrowd>**

N2304863G LEE HEECHAN

**1. Introduction**

It is impossible to talk about modern IT services except for cloud computing. As the Internet and its technologies advance, humankind has gained the power to use more resources, to store more data, and to do task (or calculate) more. Modern-day IaaS, PaaS, and SaaS products, such as Firebase, Amazon Web Service, are greatly helping various businesses realize their potential online and accelerating the development of the world.

Based on this environment, crowdsourcing can be developed more through the Internet. Crowdsourcing, a concept rooted in the collective wisdom and efforts of diverse individuals, has revolutionized problem-solving by leveraging the power of distributed networks. Beyond simple Q&A, Amazon Mechanical Turk (MTurk) have demonstrated that online crowdsourcing can be used to gather people's wisdom. ReCAPTCHA was also used for security purposes to prevent spam bots from attacking while using crowdsourcing to identify text data, or images.

Large Language Model, LLM is making a big impact on the world. Typical AI was difficult for ordinary people to custom. People are only able to use completed application wrapping AI. However, advances in various LLMs, e.g. ChatGPT, have lowered barriers to using AI, and numerous businesses are being emerged based on flexibility of LLM.

Distinctive point of LLM from other AI services is that it is easy to tune and utilize AI. LLM is tuned and used in various businesses, and its methodology is increasingly spreading. This is called prompt engineering. Users try fine tuning and setting prompt values so that LLM can give the answer in the format they want.

However, in the case of prompt engineering, people who are new to it must experience numerous trials and errors. Since it is a field that requires a basic experience and theory, I think there is a possibility that users could achieve the prompt manipulation they want based on collective intelligence. Therefore, in this project, I would like to introduce a crowdsourcing solution to find the best prompt phrase that can tune LLM according to the specific needs of users.

**2. Problem Statement**

The use of LLM is spreading widely. Therefore, many people want to use LLM to do their jobs or their own businesses. Students write reports and do assignments with the help of LLM, and startups utilize LLM into their own products.

But it's very difficult to make LLM to return the response in desired form. First of all, the big problem lies in the ambiguity of language. People may have different acceptances with the same sentence. From a machine's point of view, it can be difficult to fully understand a person's intentions, so users have to give clear instructions. But it's difficult for someone who's not skilled.

It's also not easy to know to what extent LLM understands the intentions of the user. In other words, debugging is not easy. Users must go through trial and errors in several approaches to get the results they want, and in some cases, they may not get them. This is a difficult desire for inexperienced people.

These problem lies in the lack of processes or tools for users to identify and optimize prompt statements that effectively guide to tune LLMs towards desired responses. Without guidance or support, users may face numerous trials and errors in their attempts to manipulate prompts, hindering their ability to fully cultivate the potential of LLMs in their domains. This inefficiency not only consumes valuable time and resources but also poses a barrier to widespread adoption and utilization of LLMs in various applications and industries.

Therefore, the primary challenge of this project is to **develop a crowdsourcing solution that makes users to collaboratively identify and refine prompt statements that enhance the performance and usability of LLMs**.

**3. Solution Design**

There are two main things I learned while trying prompt engineering in Assignment #2. The first was that **clear goal(output) and clear explanations were needed**, and the second was that the **model should be tested while constantly changing prompt statements and parameters**. A new service was designed with these two points in mind.

A. Overall Flow

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

*Figure 1. Overall Flow of Question and Answer*

Users can leave questions. All users can check posted questions and write their own answers about the question. When writing answers, users can check the performance of answer by testing LLM parameters and prompt statements.

Interesting point of this service is the ‘fork’ feature, which is a feature that allows users to add each one’s own ideas to generate more advanced prompts and parameter values based on the existing answers. It is also allowed for user to test their answer with GPT test during forking answer.

B. Posting Question

As mentioned at the beginning of this chapter, it is important for the questioner to present the clear goal of this LLM tuning. The advantage of LLM is that it can communicate comfortably in natural language, but ironically, the problem also arises because of natural language. It is same for communication with other users. The questioner must clearly explain what he or she wants from the answerer. Therefore, this service requires examples in question. When asking LLM, it was required to specify what type of answer they wanted. This will be of great help for the answerers to write appropriate prompts to meet the needs of the questioner.

C. Posting Answer

스크린샷이(가) 표시된 사진

자동 생성된 설명

*Figure 2. Google Cloud’s Vertex AI UI. It has prompt editor and parameter manipulator UI.*

When writing answers, we implemented it so that users can write prompts and adjust several LLM parameters. This is conceived in a UI that allows users to create prompts and adjust parameter values when using Vertex AI in Google Cloud.

D. Forking

텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

*Figure 3. Node.js Repository in Github. Successful open-source project. Forked count is 27.6k.*

Prompt engineering is based on several trials and errors. Subtle changes in prompt wording or adjustment of parameters can cause huge changes in results. Therefore, it is believed that there is enough possibility for users to develop answers that already exist. Users can get better results by forking out great answers that exist and adding their ideas. The method exists in services such as Github and Kaggle. In fact, Github is building better open-source projects through this feature, and Kaggle is using it to create better AI models based on the good work of others. Considering the characteristics of prompt engineering, forking is a very well-matched approach. Therefore, forking feature is integrated into this service.

E. GPT Test

Prompt engineering is successive iterations with changing and testing. Users need an environment where users can test quickly. Therefore, this service has added the ability to test prompts easily using the OpenAI API directly. It's a little sad that it costs real money.

**4. Solution Implementation Detail**

The service has been named <PromptCrowd>. A live demo can be found at the URL below.

Live Demo URL: <https://prompt-crowd.vercel.app/>

Github Repository: <https://github.com/Gabul99/PromptCrowd>

A. Tech Stack

The website was created with the **React.js** framework. The default language used **TypeScript**. This combination helps developers to construct a safe and efficient frontend application.

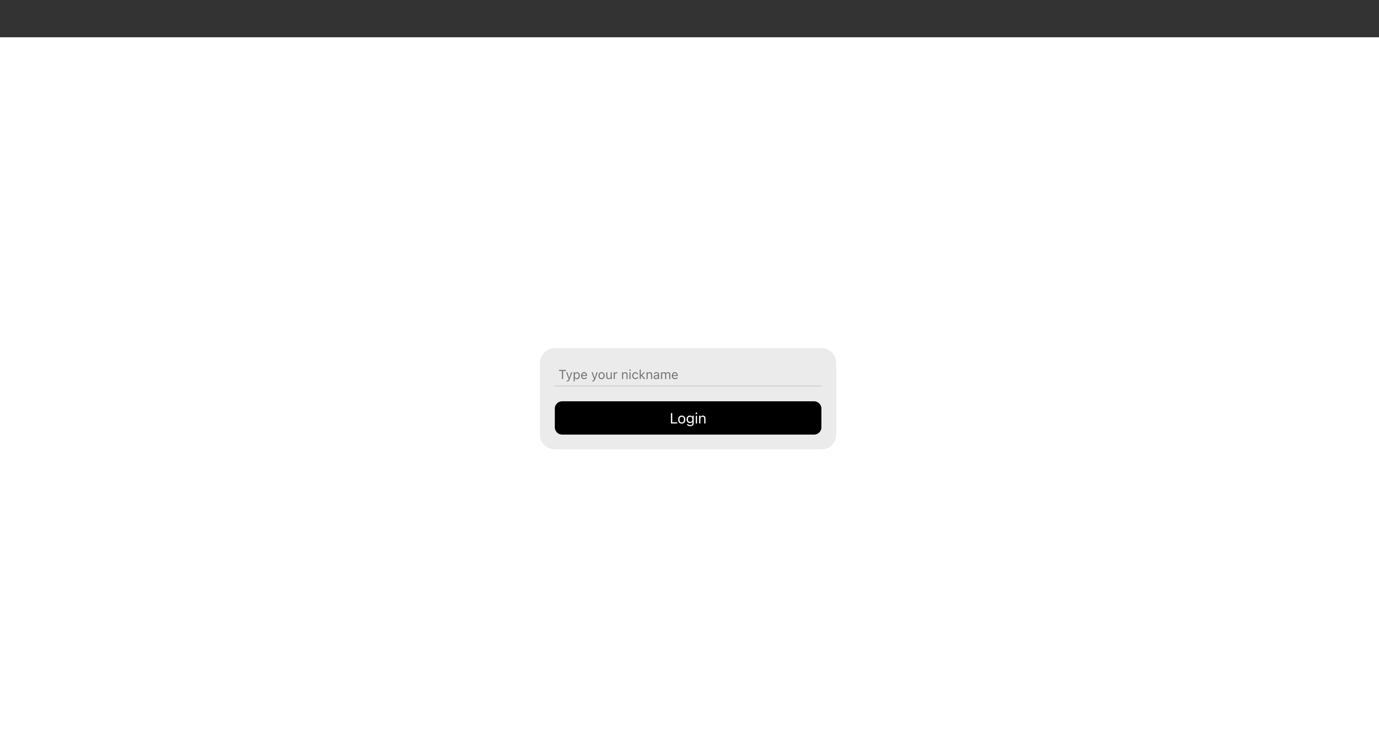
It was a solo project, and it was burdensome to organize and distribute API servers, so creating a back-end API server is impossible in this case. Therefore, <PromptCrowd> is made **serverless** using various cloud services.

<PromptCrowd> used **Firebase's Realtime Database** as its database. This application is a NoSQL environment that does not use SQL, so it loads data from the Realtime Database and manipulates it directly at the frontend side with JavaScript. Because this is not a secure way to do it, any real commercial service must either build a backend API server or use the appropriate framework for NoSQL.

For deployments, **Vercel** is selected. There are many static page distribution services, but Vercel provides the ability to integrate Github repository very easily. The security of API key is very important because it uses OpenAI API, but if you open that key on Github, your wallet will be in danger. Therefore, Vercel is adopted to manage environmental variables safely.

B. UI Walkthrough

Figure 4 shows the login screen. Since it is not an actual service, but a service for experimentation, there is no separate process such as membership registration. Users can type their nicknames and access them easily.



*Figure 4. Login page.*

텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

*Figure 5. Main page. There is question list and new question button.*

After logging in, the user is routed to the main page. Figure 5 shows the UI of the main page. Here, user can check the question list. For each question item in the list, a UI that can check how many answers are currently posted for this question.

There is a button to create a new question. If user click on that button, modal appears where user can write a question.

In Figure 6, there is a UI of the question writing modal. In this modal, there is a form for writing the title, content, examples of user messages, and answers to the expected LLM. If any of these are left blank, the question cannot be submitted. This is designed to improve the quality of the question so that the answerers can write better prompts.

텍스트, 소프트웨어, 운영 체제, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

*Figure 6. New question modal. There are 4 text inputs to have to fill.*

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

*Figure 7. Question detail page, without any selection of existing answer or new draft.*

Figure 7 shows the detail page of the question. This page is where users can check important information about the question and posted answers for the question. Users can also write new answers. All these functions are divided into three columns. In the answer list, users can check how many times each answer has been forked. It is not possible to estimate the effectiveness, but it is expected to express the reliability and quality of the answer. It is similar to judging the impact of a paper by the number of citations.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

*Figure 8. Editor UI appears when user clicks the ‘New Answer’ in question detail page.*

스크린샷, 텍스트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

*Figure 9. GPT test modal UI.*

The Editor UI, which can be seen in Figure 8, contains inputs for the title of the answer, the prompt statements, and the LLM parameters (temperature, and top-p). These are the most important and simple elements to tune on the current OpenAI API. This allows users to easily edit prompt statements and LLM parameters.

As mentioned above, small modifications to prompts and many tests are critical in prompt engineering. Pressing GPT Test on the lower left of the editor will open a chat window that lets users talk based on the prompt and LLM parameter values user have created so far through the OpenAI API. This UI is shown in Figure 9.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

*Figure 10. Answer viewer UI when user clicks the existing answer.*

Of course, users can check the answers that already exist. If users select an existing answer, they can check the UI shown in Figure 10. Users can check the prompt and LLM parameters, and in this case, the GPT test based on the answer’s prompt is also possible.

There is a fork icon at the top right of this viewer UI, and when the user presses the icon, the user can copy the content of this answer as it is and add his own ideas.

**5. How to conduct user test**

User tests are conducted on 11 university undergraduate / graduate students.

The user test was conducted online, and the instructions were provided, and the survey was conducted through Google Form.

The instructions provided are as follows.

1. Access to https://prompt-crowd.vercel.app/

2. There is no member registration, so you can log in after simply entering a nickname. If possible, for statistics, please use the same nickname when reconnecting.

3. Please post a question. You can explain by asking what function you want to create a chatbot through LLM. For example, summary, information extraction, Q&A, text classification, conversation, code generation, inference, etc.

4. Please register one answer. You can fill it out by clicking New Answer on the question details page.

5. Please Fork someone else's answer to make a better answer by adding your ideas or modifying. When viewing other's answer, you can create a new answer based on that answer by pressing the Fork button in the upper right corner.

The survey consisted of five linear scale questions and three short answer questions. It allowed users to display the UI/UX and utility of the service on a linear scale divided into five stages. It also added three short answer questions to hear more detailed opinions.

The survey questions are as follows.

1. When writing a question, did having you write examples of user messages and expected answers help clarify the question? (1~5 linear scale)

2. Was the editor writing the answer intuitive and easily accessible? (1~5 linear scale)

3. What does the fork count value representing how forked the answer is, make an impression on that answer? (Short answer)

4. Was the fork feature easy to recognize? (1~5 linear scale)

5. Do you think the Fork feature helps to make a better answer? (1~5 linear scale)

6. You can check the performance of the prompt with the GPT Test. What is your overall satisfaction with the GPT Test screen? (1~5 linear scale)

7. Do you think this service will help many users use LLM? Please write your opinion. (Short answer)

8. I would appreciate it if you could tell me what I can improve in terms of features or UI/UX. (Short answer)

**6. User Test Analysis**

During a week of user testing, a total of **11 users** used <PromptCrowd>. **Eight questions** were generated, and **13 answers** to the questions were generated. Among them, **five answers** were made using the fork function.

The user test was participated in by 11 users, out of which 7 users provided feedback through a survey.

For question 1, the majority of users (4 out of 7) rated 5, indicating that providing examples of user messages and expected answers helped clarify the question significantly. Three users rated 4, suggesting general satisfaction with this feature.

For question 2, the responses were mixed. While five users found the answer editor intuitive and easily accessible (rated 4 and 5), two users found it less intuitive (rated 2). This suggests room for improvement in the UI/UX of the answer editor.

Question 3 was a question asking how the fork count makes an impression on the user. The fork count seemed to give users an impression of the popularity and effectiveness of answers. Responses compared it to ‘academic citations’, indicating it provided a measure of reliability or quality. (“As such, the prompt gave the impression that it was more effective in achieving the desired result. It was thought to be an indicator like the number of citations in the paper.”) Some users suggested that many fork counts made them more inclined to consider using that approach or prompt. It is also able to get the response that there were implications from the perspective of crowdsourcing services and social computing services. (“It stands out first among the answers, and I think it can also function like LIKE, HEART on other community sites.”)

스크린샷, 텍스트, 다채로움, 라인이(가) 표시된 사진

자동 생성된 설명

*Figure 11. Survey results for linear scale questions.*

For question 4, the fork feature was well recognized and considered easy to use by most users (six out of seven rated 4 or 5). One user found it less obvious (rated 2), indicating the possibility of making the feature more prominent or user-friendly.

For question 5, like recognition, the utility of the fork feature was also well-received. Six out of seven users believed that the Fork feature aids in creating better answers, suggesting its effectiveness in collaborative improvement of prompts.

For question 6, the GPT test feature received generally positive reviews, with six users giving high satisfaction ratings (4 and 5).

Question 7 asked for the overall review of this service. Responses were positive about the service's potential to aid LLM usage. Users appreciated the ease of writing questions and the potential of crowdsourcing for generating diverse and effective prompts. (“I think it's a good way to get a better answer with the power of collective intelligence if you can only gather enough users.”) Some suggested improvements include more features to encourage participation, quality control mechanisms, and a rating system for forks. (“As the crowdsourcing service, I think it is better to adopt more feature to encourage user participation and ensure the quality of answers.”)

It was also able to collect opinions on various UI/UXs. There was an opinion that the overall UI would be made to be a little more distinguishable. There was also an opinion on the policy of the fork counting. If the forked answer also forked another answer, the opinion was that fork count of the original answer should also be raised. For users unfamiliar with GPT, there was also an opinion that it would be nice to have an explanation of LLM parameters, such as temperature and top-p.

**7. Discussion**

**Crowdsourcing and Prompt Engineering**

<PromptCrowd> is a project designed to solve problems through crowdsourcing that arise in the process of creating prompts for LLM users to get the results they want. I designed <PromptCrowd> to gather ideas from crowd, like Amazon's MTurk, to create better prompts.

First of all, it was meaningful to write and specify example questions and expected answers on the screen of writing questions. All users who participated in the survey evaluated that the function had a positive effect on improving the quality of the question. I was able to check that the users who participated in user test wrote down example questions and expected answers in detail. Some users commented that it would be nice to have LLM's help in the process in survey.

The Fork feature is a key feature of this service, which works well for prompt engineering and has played a good role in lowering the barriers to users' crowdsourcing participation. Prompt engineering is done through frequent testing with minor modifications. In fact, when checking answers made through Fork during the user test period, activities such as changing parameter values from existing answers or reordering prompt phrases were observed. This ensures that the fork feature allows users to get better prompt phrases with minor modifications, as intended. Even in the survey, 6 out of 7 people commented positively on the fork feature.

In the answer list, it was meaningful to expose the fork count. Users who participated in the survey responded that they felt that the fork count indicated reliability of each answer. There were responses that said they would read and refer to the contents of the answers with a higher fork count than other answers. As intended, there were also responses that mentioned the number of citations in the paper. This was an UI that was very helpful from a crowdsourcing point of view. However, there was also an opinion on the fork counting policy. There was an opinion that if the derived answer was forked, the count of the original answer should also be increased, and I agree.

Functions like these will be more effective if there are more users and more participation. It seems that the quality of answers can be improved with more diverse answers and ideas, and the problem can be solved more effectively through 'Upvote' for good answers.

The GPT test screen got nice score by the user survey. Since it uses the OpenAI API under the hood, users could conveniently test the validity of prompts without having to call the API themselves. It got the idea from SaaS such as Nemobot, and it worked well to increase user convenience.

It was unfortunate that there were no features that could further enhance the characteristics of crowdsourcing. The current features were sufficient to make prompt engineering easier, but it lacked features to engage more users and encourage communication. It's *reward and evaluation*. In most crowdsourcing services, there are rewards. In many cases, small amounts of money are offered as rewards, and systems that provide points or badges are often found. In addition, there was no indicator of the quality of the answer directly. Fork count played that role indirectly, but there were no features that could be ‘adopted’ by the questioner or 'upvote' by other users directly. These features, combined with the user system, could have managed the user's reputation, and led to community revitalization. However, this project was unable to add crowdsourcing features due to lack of human resources and technological design difficulties. If there was an opportunity, I think this could be improved in the future.

In conclusion, overall, it was able to solve the problems considered in the problem statement with <PromptCrowd>. It seems to be more effective if many users gather. In terms of crowdsourcing, there is room for development through several additional functions. Through this project, we were able to learn that crowdsourcing, cloud computing, and SaaS can contribute to bringing out the potential of LLM users and increasing usability.

**What I learned**

I've been interested in LLM usability and prompt engineering since before. I think the interaction between humans and AI is a very important issue in making software in the future, and I've been thinking about how to use LLM because LLM is the most accessible AI to the public now.

Luckily, it was able to learn about cloud computing and crowdsourcing through this class and have the experience of crowdsourcing the problems that arise from using LLM. I was able to learn a lot from the process of planning, designing, developing, and even testing users.

Through assignment #2, and experience of using prompts, I could see how users were using LLM, and it was able to design the new functions accordingly.

I was able to have a new experience in development process as well. This project was run by using the front-end, database SaaS, and deployment SaaS. In the process of implementing and distributing the project, I could feel that cloud computing and SaaS are becoming the mainstreams these days. So <PromptCrowd>’s structure is a serverless architecture. This time, it was a project I was working on alone, and since I had no experience in building and distributing back-end servers, I tried to find the most convenient way possible. In the process, I think I naturally implemented a serverless architecture, which is a very important experience for me.

And finally, it was a chance to try the OpenAI API. I didn't really use it in depth, but I was able to experience creating chatbots through implementing GPT Test UI. This was interesting, and I think it will be available to use in many places in the future.

I thought I would like to do related research in the future through this project. While I was an exchange student at NTU this semester, this project was the most interesting and enjoyable for me.