Prompting Techniques for Design Requirements of Hypothetical Application

Frank Meoni

Florida Atlantic University/GenAI Software Development Lifecycles

February 26, 2025

# Introduction

Large Language Models (LLMs) have become a great tool for accessing and delivering information. Updating your resume, writing marketing copy, and summarizing a complicated article can now be handled by LLMs. The key to their success is not that they can answer questions or complete tasks, but they have been trained to determine what you want as opposed to what you are asking. For example, if you ask an LLM “Can you please rewrite this job description to sound more technical” it will not answer “yes” which is technically the correct answer and purely logical response, it simply rewrites the description. This is also the drawback of LLMs, they are trying to guess what you need and for simple tasks it is usually capable of responding with exactly what is needed. For more complex tasks, the questions or prompt are also more complex. To this end, a new discipline of Prompt Engineering has been created to get the most out of an LLM. This paper explores several prompting methods by asking an LLM to create requirements for a hypothetical application. The responses will be analyzed to determine which prompts yield the most detailed and clearest set of requirements. There are three levels of prompts analyzed, Level 0, Level 1, and Level 2. The levels are determined by the amount of instruction and context are given in the prompt by the user to the LLM. This paper will show that Level 2 prompts yield the most detailed and comprehensive requirements.

# Methodology

The experiment is based on creating prompts to have an LLM create a set of requirements for a hypothetical application. The application is a suggestion tool for theme park attendees that uses current wait time data to minimize the wait time for attractions. This idea was chosen because at the time of writing, there are no commercially available applications that perform this function.

The base prompt is: “Please show the requirements to build an application that will take the current attraction times of a theme park and suggest the next three attractions to ride that minimizes wait times.” All the prompts in this experiment are built on this one.

These prompts will start from the simplest zero-shot, and work through several levels until finally using two Level 2 prompt techniques. The responses will be observed and evaluated in terms of how well the requirements are written. In addition, to exploring different prompts, the parameters of the prompt will be varied to explore how that changes the responses. The prompt techniques used in this experiment start with what are called Level 0, they include Zero Shot, a question with no context or additional information; Few Shots, question with some context, in this experiment there are two versions of this, Few Shots/Role Based and Few Shots V2. Few Shots/Role Based adds a prefix to the prompt that specified “you are a software developer” and Few Shots V2 uses the that prefix as well as a suffix “provide a detailed response showing functional requirements as well as assumptions regarding input by the users and UI requirements.” Moving to Level 1 prompts, Chain of Thought and what is called Refined Prompt were tested and this is a prompt that poses a question but is also prefaced by logical steps or information that helps bring context to how the response should be crafted and to align the logic with the user as opposed to an outside source (where ever the LLM gets it from). Refined Prompt is an iterative prompt that asks the LLM to refine the prompt to produce a better result. Finally, there are two Level 2 prompts, Refine Prompt Level 2 and Tree of Thought. Refine Prompt Level 2 is an extension of Refine Prompt, using the output of a refined prompt to be further refined. Tree of Thought is the most complex of the prompts and gives the best overall detailed requirements, including functionality I did not consider. This technique uses the base prompt of “Please design the requirements for an application to reduce wait times at a theme park…” and adding three different “approaches”, step by step reasoning, using real world examples, compare current solutions and choosing the best. These were then “evaluated”, by looking for the longest response (assuming longer is more detailed), and only presenting that response.

In addition to testing different prompts techniques, parameters in the prompt will also be varied to observe changes in the responses. Two parameters will be changed, num\_predict which limits the length of the response, default is 100 and this will be changed to 1000. The other parameter is temperature, which is a value that allows the LLM to be more “creative” with the response. This value ranges from 0 to 1, with 1 being the most creative and 0 the most deterministic. This experiment shows each prompt using a temperature of 0.5 and 1.0.

The LLM used in this case is Llama 3.2 using Ollama to run locally on a Windows 11 Personal Computer with a Ryzen 7 CPU (GPU is not used in this case). The prompts are coded in Python and the github library is referenced in the Reference section.

The prompts are for the same goal, write requirements for the creation of an application to reduce wait times in a theme park. The analysis will compare each response to the rest of the responses and rank them based on completeness are all parts of the application considered data sources, devices, platforms, etc. Rank will be reduced if hallucinations or issues like Semantic Drift are found.

# Results and Analysis

The first results quickly revealed that a num\_predict value of 100 is too low and cuts off the results. Before all the original batch of tests were run with the default parameters, this value was changed to 1000 so the entire response could be displayed.

As the prompts became more complex, moving from Level 0 to Level 2, unsurprisingly the time to run each prompt increased as seen in Table 1 below.

|  |  |
| --- | --- |
| Technique (num\_predict, temp) | Run Time |
| Zero Shot (100,1.0) | 14.44s |
| Zero Shot (1000,1.0) | 47.64s |
| Zero Shot (1000,0.5) | 88.24s |
| Few Shots/Role Based (100,1.0) | 18.01s |
| Few Shots/Role Based (1000,1.0) | 56.05s |
| Few Shots/Role Based(1000,0.5) | 56.63s |
| Few Shots V2 (100,1.0) | 16.17s |
| Few Shots V2 (1000,1.0) | 38.93s |
| Few Shots V2 (1000, 0.5) | 68.35s |
| Chain of Thought (1000, 1.0) | 55.67s |
| Chain of Thought (1000, 0.5) | 161.02s |
| Refined Prompt (1000, 1.0) | 240.27s |
| Refined Prompt (1000, 0.5) | 156.20s |
| Refined prompt level 2 (1000,1.0) | 266.57s |
| Refined prompt level 2 (1000,0.5) | 156.19s |
| Tree of Thought (1000, 1.0) | NA |
| Tree of Thought (1000,0.5) | NA |

Table 1

The code to update the time for the Tree of Thought prompt didn’t work correctly so those numbers are not available.

The trend for times tends to be the more complex the prompt (level 1 vs level 0) the longer it takes. In addition, Table 1 also shows that as the num\_predict is increased the longer it takes to run and lowering the temperature also takes longer to run. This is most likely since a lower temperature means the LLM is forced to be more accurate.

The analysis of the prompt responses are subjective to a point, however, the objective is to find requirements that are not only comprehensive but more importantly include requirements that are not expected. As the prompts became more complex, the responses were correspondingly more detailed. However some of the prompts with the temperature set to 1.0, sometimes showed signs of Semantic Drift. This is the case when the LLM gets “off topic” and the response shifts to a different subject. It still sounds correct, but it addresses a different question that wasn’t asked. In other examples the response has code that doesn’t seem to be related to the question or the application.

The most comprehensive response by far was the Tree of Thought with temperature of 0.5, the response included the use of chatbots, gamification, and in app challenges to spark interest and that wasn’t mentioned in any of the other responses. There were other prompts that did have detailed requirements however, they did not include new ideas.

# Conclusion

LLMs are a great tool to use for information gathering and dissemination. Even though they are good at determining the intent of the user, there is still a need to structure the prompts or questions to receive effective feedback. Prompt Engineering is a budding field that creates techniques for asking LLMs questions, so the responses are valuable.

The experiments conducted in this study range from basic Zero-Shot prompting, which lacks contextual information, to advanced Tree-of-Thought prompting, which incorporates multiple layers of structured prompt. As the prompts increased in complexity so did the responses. One area of concern is in the tests where the temperature is set to 1.0, there was semantic drift in several of the responses, completely off topic from the question. In this case of creating requirements for a hypothetical application for minimizing wait times at a theme park, the Tree of Thought prompt with a temperature of 0.5 yielded the best response. This was the only prompt technique that produced innovative functions for the application.

LLMs hold a lot of promises as a powerful tool for the many jobs like summarizing complex articles, breaking a process down to manageable pieces, and writing marketing copy. However, knowing how to ask the questions will be the key to maximizing the potential of Large Language Models in the future.

# Future Work

Future work in this study can be focused on many areas and it’s important to test each part on its own. For example, test zero shot by using the same prompt and vary each parameter (like temperature) on its own then jointly with another parameter to explore the variation in responses. In addition, there are many other knowledge domains that can be studied such as medicine and tax law.

# References

All the code as well as the prompts and responses to each Prompt Technique can be found here: <https://github.com/frankpmjr54/GenAI.git>.