https://doi.org/10.65091/icicset.v2i1.7

AI-Powered Roadmap Generator Prompt Engineering

Amit Gupta
Manipal University Jaipur
Jaipur, India
amit.2424150015@muj.manipal.edu

Vikash Kumawat
Manipal University Jaipur
Jaipur, India
vikash.2424150052@muj.manipal.edu

Niyaz Ahmad Wani
Department of Computer Applications
Manipal University Jaipur
Jaipur, India
niyaz.wani@jaipur.manipal.edu

***Abstract*—One of the most common issues in AI is discussed in this paper the problem of making Large Language Models (LLMs) produce the correct, clean and structured output in the form of a JSON. This is of significance in a majority of real world applications employing AI. The researchers examined the solution of this issue by an app named "Roadmint". They searched the key working formula. The trick is simple like - Make the artificial intelligence know what to do, Clearly give the JSON structure inside the prompt, Give a few good examples, Create some strict rules - such as what's the AI is not allowed to do. The most appropriate is that everyone can use this approach as a template. In case you are developing an AI project and you want your model to provide the same and correct output, such a strategy can make your system more stable and easy to manage.**

***Index Terms*—Prompt engineering, Natural language processing, Artificial intelligence, Machine learning.**

I. Introduction

During the last several years, Artificial Intelligence (AI) has become an extremely fast-growing movement, primarily due to large pre-trained language models (LLMs). These models have already transformed the approach to the concept of Natural Language Processing (NLP) entirely because they can address numerous tasks simultaneously.

In order to make these models useful in particular tasks, there has come a new technique which has gained significance to make these models better in performing particular tasks, which is known as the prompt engineering. [6] [7] It simply happens to be the art of good commands (prompts) to the model - without altering its inner settings - so that it provides the sort of output we desire. It finds application in most areas such as medicine, finance, IT, and law.

Compared to the previous strategy known as a fine-tuning, prompt engineering is much simpler and less expensive and requires a large amount of computer power and huge datasets to retrain the model. Due to this reason, real-time engineering has become one of the core elements of the development of AI. It makes AI systems more versatile and applicable to all types of work such as responding to questions, logical reasoning, writing code, or even coming up with stories and poems.

A. The Last Mile Problem: Constructive Data Generation.

Since Large Language Models (LLMs) are no longer seen as a mere research project, but as a valuable component of business software, they no longer need to write just text that is fluent and written like a human being. They also now have to generate machine-readable structured data which can be easily interpreted and consumed by software frameworks.

This capability is the one that serves as the so-called last bridge, that is, it provides the communication between the LLMs and other software components, including APIs and third-party programs, and ensures that all other components co-exist as a single system. Such systems are referred to as Compound AI Systems in which the LLMs are connected to a range of tools and calculators to address more complicated problems. [2]

In such an arrangement, the role of **JSON** (JavaScript Object Notation) is very critical. Due to its simplicity, readability, and compatibility with most programming languages, JSON has become the default data interchange format, web API, configuration file, and inter-microservice communication format.

Hence, when an LLM is unable to produce right and correctly structured JSON, it is a severe constraint. The fact that one can generate valid JSON is therefore not merely an additional capability but a must-have skill to use LLMs in a wide variety of practical uses such as automatic data extraction, content management systems, and multi-agent AI systems.

II. Related Work

This section presents a systematic review of the evolution, the underlying principles, and the advanced techniques of prompt engineering, and serves as a theoretical foundation for the case study.

A. The Evolution of Prompting

The research field of prompt engineering has evolved within the domain of Natural Language Processing (NLP) from an early development of rule-based systems to now more complex data-driven systems. The beginning of modern view began in 2020 with the groundbreaking large-scale pre-trained language

model OpenAI's GPT-3, moving some of the effort away from fine-tuning models and placing effort and emphasis on developing efficacious prompt structures to achieve the desired result. [1] Since GPT-3, the field has grown quickly, effective with prompt engineering by developing more complex and thorough sets of instructions, and now effective prompts are an important piece of maximizing the capabilities of the modern large-scale pre-trained language models (LLMs).

B. Foundational Techniques: In-Context Learning

The simplest form of prompt engineering is In-Context Learning (ICL) that uses prompt-based examples to learn. [4] [5]

- Zero-Shot Prompting: This is a simple form of interaction that completely depends on the inner world of knowledge of the model without examples. This is efficient in simple case of prompts, but will soon collapse as prompts become more complex, and in case of generated output.
- Few-Shot Prompting: This is a form of prompt engineering where the prompt contains a small number of examples of input/output - this will be made to work in other more complex or domain-specific prompts because this will aid the model in knowing what is intended to be used and what forms and styles will be used.

C. High-tech Reasoning and Decomposition Methods

Prompt engineering research is also active in getting complex reasoning out of LLMs.

- Chain-of-Thought (CoT) Prompting: This type of prompting is a transformative method of prompting that encourages the LLMs to provide reasoning in stages, and thus, they are more precise on multi-step problems. [10] It has zero-shot or few-shot application.
- Variants and Extensions: CoT influenced methods such as Self-Consistency (creating multiple reasoning paths and choosing the most common response) and Tree-of-Thoughts (ToT) and Graph-of-Thoughts (GoT) (creating multiple reasoning paths at once and determining which are making progress and which are not). [9]

D. Prompt Design Principles of Consistency and Control

In practical prompt engineering, the main goal is to reduce the confusion and make the output more predictable. Short prompts are also token-efficient, where as detailed prompts are required to obtain desire results in certain areas, such as in structured data generation.

- The Importance of Constraints and Clarity: The requests should be very specific and clear. The explicit negative constraints prevent the common failure modes(what the model should not do).
- Persona and Role-Playing: Giving the LLM a certain persona will dictate its tone, style, expertise, and behavior, and shape the output in accordance with the context. [11]
- Model-Agnostic Design: It is a design that builds prompts that can be applied to any LLM by operating with standard formats and instructions, which offers flexibility and power to go future-proof.

E. State of art in Structured Data Generation

Production of structured data is a major difficulty in prompt engineering. Research indicates that increased structure and clarity (especially in the few-shot examples of the desired JSON structure) significantly enhance the validity and consistency of the generated output. In the case of structured data tasks, having a well defined and detailed prompt can be the only guarantee of reliability in a production environment. [8]

III. System Architecture

This part shifts the theoretical contexts of prompt engineering to its practical application by introducing a in-depth examination of the prompt architecture that is proposed to drive the application of the prompt architecture in the Roadmint Application.

A. Overview of the Roadmint Application

Roadmint is a web-based application that enables users to be offered, AI-assisted learning roadmap to any topic of their choice such as Web Development, Data Structures and Algorithms, or Machine Learning. Its main advantage is to just enter any desire subject into Roadmint, like Web Development, Data Structures and Algorithms, or even Machine Learning, and receive a day-by-day learning Roadmap. The app focuses on an cleen ui and simple user experience to encourage a focused learning experience and has features like customized daily steps, tracking of progress by marking every topic as learned and exporting the whole roadmap to a CSV file. The fact that it promises a high-quality, detailed and personalized roadmap with only one topic input shows the power and reliability of the LLM prompt engineering. The success of the application directly relies on the capability of the prompt to produce a response that is constant, logically consistent and structurally valid to be analyzed and represented by the front-end application.

```
{
  "Category": [
    {
      "subtopic": "string",
      "difficulty": number
    }
  ]
}
```

Api responce Structure

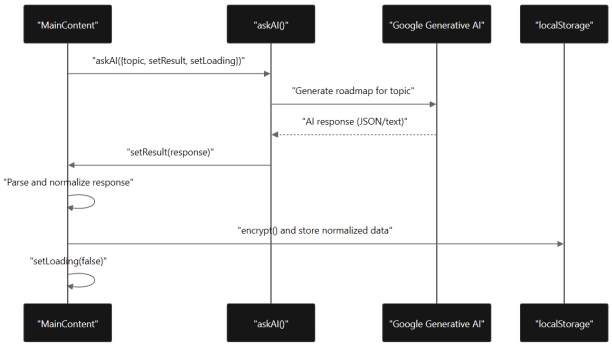

Fig. 1. Proposed structure of the Roadmint prompt system.

B. Architectural Deconstruction of the Core Prompt

Although the actual code of the Roadmint prompt is proprietary, it is sufficiently standard and high-quality to make it possible to reverse-engineer the probable architecture of the Roadmint prompt using the best practices that have been established and reviewed in the preceding section. The prompt itself is postulated to be a complex multi-component structure, with every component having a specific role in steering the LLM to the desired output.

C. Component 1: Role and Goal Definition

The urgency virtually commences with establishment of a definite and commanding role of the LLM. It is one of the techniques that prime the behavior, tone, and domain of knowledge of the model.

Hypothesized Implementation:

"You are a knowledgeable curriculum developer and high-level technical mentor. You are expected to design a detailed, rational and useful learning map of a particular technical subject. The roadmap must be designed to show a learner his or her way beginning at the basic concepts to the more sophisticated applications."

This first command immediately contextualizes the task, telling the model to be an expert (regulate the quality of content), and actually specifies the ultimate output form requirement (a single JSON object).

D. Component 2: Context and Input Processing

The immediate should then specify the manner in which the variable user input should be treated and that is the subject of the roadmap. In this section, the context about the content generated is given and the scope of this content is established.

Hypothesized Implementation:

"The user is going to input a topic. What you need to do is to create a comprehensive learning roadmap on this topic: topic. The roadmap must be appropriate to an enthusiastic novice who might have some general knowledge of the technical aspect without having specialized knowledge of the subject offered. Make sure that the roadmap addresses all the necessary sub-topics in a logical sequence."

This element connects the abstract purpose to the tangible input by the user and defines the target audience (a beginner) that assists the model in adjusting the depth and complexity of the material.

E. Component 3: Task Decomposition Instructions (Implicit CoT)

In order to make sure that the created roadmap is not simply a haphazard set of keywords, one can expect the prompt to have instructions that drive the model through a line of reasoning prior to it starting to give out the final JSON. This is reminiscent of a Chain-of-Thought process to impose rational consistency. The hypothesis suggests that the effects of a training program on PR staff can be determined by analyzing their performance records and interactions with management and supervisors by human.

Hypothesized Implementation:

The internal steps to be followed before building the JSON are:

- Determine the main pillars and the main notions of the topic. Determine the main pillars and the main notions of the topic
- Arrange these ideas into a set of logical modules, the first one being the fundamentals then moving up to the more advanced or specialized subjects.
- In every module, divide the content into small parts and use these parts as bits of daily learning. Every task must concentrate on one clear and definite concept or ability.
- Each day, prepare a brief yet informative list of the description of the tasks specifying their importance and what a learner will get at the end.

Such directives coerce the model to do a pre-computation of the structure of the roadmap, to give final output, which is pedagogically correct.

F. Component 4: The In-Prompt JSON Schema

This is perhaps the most important element in the provision of structural validity. The specification of the necessary JSON structure is given in detail and with comments in the prompt. This provides a clearcut, unquestionable template that puts a lot of limitations on the output format of the model.

Hypothesized Implementation: The final output should be in the strict conformance to the following JSON schema. Do not depart out of this form.

```
JSON
{
  "roadmapTitle": "string",
  "totalDays": "number",
  "modules": {}
}
```

This in-prompt schema is one of the strong guides, as it gives names of the key, type of data, level of nesting, and even comments to define the intention of each field.

G. Component 5: Few-Shot Example

To complete the abstract schema, there is a concrete example which gives an effective demonstration of what is wanted. It is a traditional a few-shot prompting method, which enables the model to fix its ideas of the necessary form and content style.

Hypothesized Implementation: The following is a sample of the valid JSON response to the topic of Learning Git Basics:

```
JSON
{
  "roadmapTitle": Git Basics,
  "totalDays": 3,
  "modules": {}
}
```

This instance gives a full-fledged model which the model can directly copy.

H. Component 6: Negative Constraints and Formatting Rules

Last, the prompt has some stern, unambiguous guidelines on what should be avoided. These constraints are aimed at removing the most frequent failure modes, i.e., the use of non-JSON text.
Imposed Costs: Within the framework of the new payment system, the hospital faces extra costs due to the need to regularly obtain consent, trainings, and testing kits to assess medication errors.

Hypothesized Implementation:

```
CRITICAL RULES:
- Your answer should have the form of { and }.
- No text, explanation, apology or markdown
    (such as json) should come before or
    after the JSON object.
- This whole output should be a single,
    unprocessed, and fully parsable block of
    JSON.
- Make sure that all the strings are escaped.
```

These concluding decisive commands are aimed to fix the desired behaviour and avoid the chatting nature of the LLM that may have poisoned the formalized text.

IV. Methodology

The proposed structure of the Roadmint prompt is not just a set of tips, but a structured engineering project which was created to generate trustworthy, structured output. The following section is an analysis of the elements of the prompt functioning in synergy to bring about this consistency being linked to the practical application to the theoretical principles of prompt engineering. The success of the prompt may be interpreted as a sort of in-context fine-tuning which makes an intensive, task-specific context available at the inference time in order to induce a temporary, specialized model behavior without modifying the weights of the underlying model. Such a design bypasses the resource-intensive process of traditional, model-specific fine-tuning and, operating by providing a rich context, including a role, schema and examples, creates a strong local optimum in the generative probability space, where outputs that match the specified structure are much more likely to be generated than the rest. [12]

A. Evaluating Structural Integrity and Combating Failure Modes

Every member of the Roadmint prompt specifically addresses the most frequent failure modes that are found in literature on structured data generation. The commonest failure mode is the creation of syntactically invalid JSON. This is defended by the combination of explicit In-Prompt JSON Schema and concrete Few-Shot. The schema serves as a structure specification, specifying the keys, data types, and nesting layout that is required and the example gives the model a concrete template to emulate. They all significantly minimize the likelihood of syntax errors, including omitting commas or brackets being placed in the wrong places, or incorrect or invalid key-value pairs.
The other typical failure is that conversational filler, or some other non-JSON text, has been included, which cannot be programmatically parsing. The Negative Constraints and their direct and emphatic guidelines (e.g., *Your response MUST include { and must include a final }*), are actually meant to prevent this behavior. This is also supported by the original Role and Goal Definition, that makes the task of the LLM that of a data-generating system as opposed to conversational partner, thus minimizing its propensity to generate conversational artifacts. This combination is a succession in ensuring that the model has to produce only the raw object in the form of a JSON, which can directly be consumed by a downstream application.

B. Mapping Theory to Practice: A Principled Approach

The efficiency of the Roadmint prompt is no coincidence; it is the fruit of the introduction of proven postulates of prompt

engineering in a single-minded way. The table below breaks down the architecture of the prompt and relates each element with the academic principle behind the element and its exact impact on the output.

TABLE I
MAPPING THEORETICAL PROMPT-ENGINEERING PRINCIPLES TO PRACTICAL ELEMENTS IN THE ROADMINT PROMPT.

Component	Implementation in Roadmint Prompt	Academic Principle	Observed Effect
Role Definition	"You are an expert curriculum designer and senior technical mentor..."	Persona Pattern / Role-Playing	Guides tone, style, and technical accuracy.
Task Decomposition	"Before constructing the JSON, follow these internal steps: 1. Identify core pillars... 2. Organize into modules..."	Implicit Chain-of-Thought (CoT)	Enforces logical structure before formatting.
Schema Specification	A detailed, commented JSON structure defining all keys, types, and nesting.	Constrained Output / Schema Priming	Restricts format, reducing syntax errors.
Few-Shot Example	A valid JSON output for "Learning Git Basics".	In-Context Learning (ICL) / Few-Shot Prompting	Demonstrates target format and reinforces schema.
Negative Constraints	"CRITICAL RULES: Do NOT include any text... Your response MUST begin with {..."	Explicit Constraints / Guardrails	Prevents conversational filler and enforces JSON-only output.

The table serves as the primary analytical evidence for the case study. It gives clear evidence and organization to help support the thesis of the paper, by showing (1) that each part of the prompt has an identifiable aim, (2) a firm theoretical basis, and (3)

C. Quantitative Evaluation of Structural Validity

Besides qualitative analysis, the quantitative assessment was performed to assess the effectiveness of the proposed prompt architecture to produce valid structured outputs. The testing was done on syntactic correctness and conformance to schema of the generated JSON responses. In every single experiment, 100 independent generations were generated at a constant input topic (Web Development Roadmap) and with the same parameters of generation. All the outputs were automatically checked with the help of a JSON parser. A response was deemed a valid one when it met all the following conditions:

1) syntactically valid JSON,
2) compliance to a specified schema structure
3) absence of any non-JSON text.

To evaluate the value of various prompt components, the proposed Roadmint prompt was contrasted with two strategies used as the basis:

1) a native prompt of no structural restrictions
2) a schema only prompt that had a JSON template but either no role definition, no task decomposition, no few-shot examples or no negative constraints.

The findings show that there is a significant increase in reliability. A naive prompt got 61% valid JSON responses, the schema-only prompt got 84% and the proposed multi-component Roadmint prompt got 100% valid JSON responses on 100 generations. These results are empirical proofs that schema priming, coupled with implicit chain-of-thought-style reasoning, small sets of examples, and rigorous formatting restrictions greatly decrease the prevalent cases of failure on structured data generation.

D. Qualitative Analysis of Generated Content

The quality of the output produced in terms of the generation of valid JSON is not all; it is the generation of high quality JSON.
Here the Task Decomposition Instructions is important. The prompt provides a instructional framework upon the content generation process by asking the model to first find core concepts and then categorise these concepts into modules, then further subdivide them into tasks that can be used on a daily basis. This does not allow the model to just enlist associated words but also promotes the model to develop a curriculum that has a logical flow and progression.
In addition with this the elaborate descriptions of the In-Prompt JSON Schema for example a detailed 2-3 sentence explanation of the topic of the day and learning objectives are micro-prompts of each content. These are comments indicating the preferred length, depth and use of the string values generated by model. This takes care of the fact that the descriptions are more than placeholders but actually informative and consistent with the learning objectives. A combination of all these has a final output which is not only well-formed, but comprehensive, logically organized, and instructional useful the main features of the Roadmint application.

E. Cross-Model Robustness Analysis

In order to assess how the model-agnostic nature of the proposed prompt architecture is, the identical experimental setting was applied to several large language models. Roadmint was run on GPT-4, Claude, and Gemini using 100 generations each.

The timely showed excellent structural validity in all the assessed models. The GPT-4 generated 100% valid JSON, Claude generated 94% and Gemini generated 92%. Although these models may have architectural and training differences, the findings indicate that the immediate design principles used in Roadmint are generally applicable to LLM families, which validates the argument that the method is non-model-specific but rather general.

V. FUTURE RESEARCH

Although the suggested prompt architecture shows a high level of structural reliability, the current research is confined

to a limited number of input topics and regulated generation parameters. The subsequent research will include mass automated assessments in different fields, varying prompt length, and varying temperature conditions. As well, execution-time validation and self-correction systems can also be incorporated to enhance resilience in the manufacturing world.

- Persona-First Principle: It is always important to start the prompt by giving the LLM a defined and professional role related to the data being produced (e.g. You are a database administrator, You are a financial analyst). This instantly guides the model and prepares the model to conduct a technical, data-oriented work.
- Ask the Structure Upfront: The first lines of the prompt clearly state the main rule that the answer should be in proper JSON object. This sets the most important condition from the very beginning.
- Decompose Before Composing: Include: Gives a sequence of step-by-step instructions, which the model will follow the process steps, before it makes the final structure. This implicit CoT will provide logical integrity in the data content.
- Give a Full and Commented Schema: The entire JSON schema is typed in the prompt and it contains all keys, the expected data types (string, number,boolean,array, object) and the nesting level. A comment can be used in the schema to explain the objective and limitation of each field.
- Use Canonical Instance: Use small, but clear few-shot example of a good JSON output. This example clearly to be representative of the desired structure and the content style, it will act as a clear guide for the model for desired output.
- Lastly, Strict, Final Guardrails: Finish the prompt by a sequence of restrict regulations. These should prohibit any textual data of dialog or markdown format other than a JSON object and reiterate the fact that the output is supposed to be a raw, easily parsable Content.

By following this six-part architecture one can offer a solid design to achieve a great deal of reliability in structured JSON generation across a broad set of LLMs and applications.

VI. Conclusion

The goal of this research paper was to find an answer to a crucial challenge, which is the ability of Large Language Models (LLMs) to generate sound and well-organized information. This has continued to be one of the greatest challenges in creating trustworthy applications that are AI-centric. It evaluated one of the most effective prompt designs in the Roadmint application, which demonstrated that a multi-layered and thoughtfully designed prompt engineering strategy has the capability to decrease the inherent heterogeneity of LLMs and enables them to produce consistent and high-quality outputs in the form of JSON.

It was found that the effectiveness of the prompt was not due to a single trick. Rather, it was a combination of multiple essential factors such as giving the model a distinct role giving it instructions on how to reason, having a prominent and annotated JSON schema as part of the prompt and having strict rules to prevent the production of unwanted text. This prompt structure similarly serves as a combination of in-context fine-tuning that provisionally modifies a general purpose model to achieve a particular task, in this case roadmap generation, during inference.

The key finding of this study is that it is not a coincidence when generating reliable structured data is the task of engineering but it is not a impossible task. Developers can minimize the inability to predict AI results and make them less random and predictable by implementing more elaborate prompts and going beyond instructions. The strategies, recognized in the case study of the Roadmint may be applied as a practical guideline to the developers who wish to develop more foreseeable and reliable AI systems. As this paper demonstrates that the timely engineering practice will feature prominently in ensuring the LLMs safely and effectively become a part of our technology.

References

[1] 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.

[2] Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023.

[3] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729, 2024.

[4] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[5] Clyde Highmore. In-context learning in large language models: A comprehensive survey. *Preprints*, 2024.

[6] Haochen Li, Jonathan Leung, and Zhiqi Shen. Towards goal-oriented prompt engineering for large language models: A survey. *arXiv preprint arXiv:2401.14043*, 2024.

[7] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[8] Connor Shorten, Charles Pierse, Thomas Benjamin Smith, Erika Cardenas, Akanksha Sharma, John Trengrove, and Bob van Luijt. Structuredrag: Json response formatting with large language models. *arXiv preprint arXiv:2408.11061*, 2024.

[9] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[10] 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, 2022.

[11] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

[12] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.