Optimizing LLM prompts for Automation of Network Management: A User's Perspective

Vishnu Komanduri, Sebastian Estropia, Scott Alessio, Gokhan Yerdelen, Tyler Ferreira, Geovanny Palomino Roldan, Ziqian Dong, Roberto Rojas-Cessa

***Abstract*—Although being linguistic assemblers and decomposers, large language models (LLMs) have found their use in analysis and generation of code, art, and design of electronic circuitry, industrial parts, and network design. However, the probabilistic nature of generative LLMs makes network design and implementation scenarios prone to errors. The complexity levels of design may exacerbate the number of inaccuracies included in an LLM's response. Therefore, it is necessary to identify the features that make prompts generate effective and error-free responses as users. To reduce the error rate of the responses, we test prompt specificity in text and schematic descriptions. As various degrees of specificity, we compare highly intuitive to highly specific prompting. The responses are expressed as network schematics and router configuration commands that are evaluated with our proposed scoring policy. Our tests include networks with three levels of complexity and multiple levels of specificity in text and graphic prompts. The results show the trade-offs on the text and graphic modes and the degrees of specificity.**

I. Introduction

Natural Language Processing (NLP) has made significant progress in recent years, going from simple yet effective text analysis and optimization tools to Large Language Models (LLMs), which boast the capabilities to solve complex problems using a vast and dynamic array of knowledge [1]. Such models, which have largely been associated with general-purpose tasks such as drafting personal and professional letters, or summarizing passages from books, have recently grown increasingly capable of problem solving and design. Their remarkable success in such tasks, have generated a large demand for applying them to a variety of industrial applications [2]. In these domains, LLMs are being sought for the design of electronic circuits, mechanical design, art, and implementation of communications infrastructure [3].

Network design and implementation is an application in which LLMs can be used as an accelerating tool to automate the design, management, and configuration tasks [4].

While LLMs are quite effective at generating expeditious data according to an end-user's rules, they also pose subtle but definite risks when applied to network design, optimization, and modification. The speedy responsiveness of LLMs is significantly counterbalanced by the inclusion of errors and ambiguities in their responses, rendering LLMs promising but also ineffective in real practice. From a high-level design user, one can identify that LLMs typically perform two operations: First, they strive to identify the user's objective(s) in their prompt to clearly identify the problem. Second, LLMs aim at generating an applicable solution to the presented problem to fulfill those objectives; albeit at some extent. It must be noted that successfully performing both of these tasks is quite challenging, as defining the problem in some cases is typically just as challenging as addressing it, even for people. It is important to recognize that while we, as users, may not have control over how an LLM solves a problem, we might have control over facilitating the LLM identify the objective of our request.

In this paper, we aim to answer the following question: What is the input prompt feature that plays a crucial role in the correctness of a generated response for network implementation? Answering this question would help us to use LLMs more effectively. To address it, we tested three approaches to LLM prompts for network design and router configuration. Therefore, we introduce a scoring policy to evaluate the LLM response's correctness and performed extensive experimentation with various prompts and evaluated the LLM responses. We tested a direct approach where the input prompts are categorized by the level of specificity and complexity in the network parameters.

Our designed scoring policy is based on identifying inaccuracies, where incomplete, incorrect, and duplicate information accrues for negative points. The prompts are tested with three levels of network complexity [5] and up to ten levels of specificity. Furthermore, we tested two types of prompting: text and, our proposed approach, schematic. We hypothesize that schematic prompting is more effective for network design, as LLMs easily identify the conditions of a network and the objective of the prompt. Response assurance requires analyzing the performance of LLMs in typical networking scenarios and dives into identifying where responses fall short, and showcases their intrinsic randomness that potentially both helps and hampers the model's performance. Our results highlight the delicate balance between handling information and the lack of training in reliable design by LLMs. These results also showcase that specific language and network schematics have the potential to increase the overall correctness of generated LLM responses and further evaluate the strengths and weaknesses of various general-purpose LLMs on network design and implementation.

The remainder of the paper is organized as follows. Section II discusses related work on using LLM for network design

V. Komanduri, S. Estropia, S. Alessio, G. Yerdelen, T. Ferreira, G. Palomino-Roldan, and R. Rojas-Cessa are with New Jersey Institute of Technology, Newark, NJ 07102. Z. Dong is with New York Institute of Technology, New York, NY 10023. Email: vk379@njit.edu, rojas@njit.edu

and implementation. Section III discusses the variations of different input prompts proposed and tested in this paper, and also a comparison of responses obtained by various LLMs. Section IV presents our evaluation results. Section V concludes the paper.

II. Related Work

Network implementation is often complex, where even slight intricacies in methodology can make a significant impact on the overall system [6]. The application of verification technologies to the networking domain has proven quite challenging, with a majority of state-of-the-art approaches relying on rule-based verification methodologies or direct human intervention [7]. Rule-based verification is tedious, often times requiring a vast set of rules to cater to the wide-variety of responses that a model can provide. Human-intervention, which is typically referred to the definitive solution to most LLM verification problems, greatly undermines the model's overall automation capabilities. Text, which is currently the preferred medium of expression used by LLM, is often times subjective, making verification ambiguous. For these reasons, recent work has proposed innovative approaches to tackle the problem of content verification by converting text-based responses to alternative representations to effectively evaluate them. Besta et al. [8] proposed the use of word-embeddings to quantify a text-based response into a vector for analysis and comparison with alternative vectors. The approach determines the overall stability of a response by exercising multiple iterations. Similarly, Sun et al. [9] proposed a graph-based approach in which a response is encoded graphically to visualize a response and analyze its effectiveness. Visualizing the responses of LLMs in alternative mediums reduces the ambiguity that spawns from text, and helps to establish specificity.

Although response verification is an extremely integral part of an autonomous network implementation system, it also fails to prevent the generation of invalid, incorrect, or incomplete responses. Accurate response generation for networking scenarios is difficult, in part due to the lack of LLMs dedicated to networking and also to the inherent randomness of LLMs, which in some cases leads to hallucinations (i.e., with apparent disassociation with the request or merely incomplete) [10]. For this reason, zero-touch network and service management (ZSM) systems typically need additional support from additional techniques, such as prompt engineering and domain-specific training [11]. NetLLM is one such approach that provides network-specific knowledge to an existing general-purpose LLM to make it adaptable to a wide variety of network scenarios [12]. This style of approach greatly enhances the effectiveness of an existing model, without the need for the creation of an entirely new one. Ifland et al. [13] similarly designed an entirely new LLM for networking purposes built on top of OpenAI's general purpose ChatGPT.

While these approaches are quite powerful, there has been limited research on prompt-engineering techniques that leverage network implementation and immediate use of existing public LLMs. Here, we address this gap by analyzing various prompts for network design and implementation elaborated with various levels of complexity and specificity. Moreover, we also propose an optimizing approach to increase the response success rate based on using network schematics.

III. Prompt-Engineering Methodologies

Prompt engineering adds the ability to bolster the content-generation aspect of an LLM by reducing ambiguity and increasing clarity, so that the model can effectively understand the intentions or expectations of the end-user. We theorize that for any input prompt, there are many variations of it because of the intrinsic diversity in natural language, but with some being more effective than others for LLMs. Therefore, maximizing the accuracy of a response with prompt engineering requires two factors: First, the means to convert the initial prompt into an ideal one; which is a prompt that obtains the LLM's response with 100% accuracy, and the second is a methodology to consistently and effectively evaluate the accuracy of the received response. Therefore, it is essential to identify the leading factors that generate the correct responses when developing an input prompt, as well as to verify the received responses and validate their performance.

A. Prompt-Conversion

From a high-level perspective, an ideal input prompt, which is intended for network implementation purposes, contains three distinct characteristics: a clear description of the network elements, a strong set of deliverables, and a list of commands that need to be provisioned in each element. It must be noted, that this combination would reduce the LLM's by-design reliability on the data it has been trained on, and instead would cause it to utilize data provided by the end-user. Although this is quite ideal in theory, providing the actual values is akin to providing the solution to the network-design or implementation problem, as a majority of the labor involved with the design is performed by the end-user, leaving the LLM simply regurgitating existing information in a format expressed by the deliverables. Additionally, asking the user to design an ideal prompt further diminishes the automation capabilities of the LLM and adds overhead to the overall efficiency of the model. Therefore, we believe that expecting the user to come up with such a prompt is not practical for zero-touch network implementation purposes, rendering the user to focus on alternative methodologies.

As an ideal-prompt is not practically feasible, we propose an alternative version of it. In a definition more tailored towards network implementation, the optimal prompt is one that would clearly define the necessary network elements, contain a correct and complete set of deliverables, such as configuration scripts, network description, or network schematics. While a partial solution is not always feasible in practical scenarios, we argue that introducing a partial solution, regardless of how minor it may be, has the potential to steer the model in the right direction to accurately meet the expectations of the end-user. In lieu of a partial solution, a set of explicit requirements proves sufficient as long as the model can clearly understand the expectations. There exist quite a few pitfalls in natural language which prove harmful to the clarity of LLMs

prompts. They are described in Table I. If an optimal prompt is considered to have a rating of 100% specificity and clarity, a shortcoming in a response would accrue for negatively points in the score, consequently reducing the correctness of the response. The minimum score is zero.

TABLE I: Prompt-Design Pitfalls.

Pitfall	Penalty (%)
Unclear definitions	50
Word count < 50	25
No explicit IP / Subnet Addressing	25
Grammatical errors	15
Redundant phrases	5

In addition to the existing prompt-engineering methodology, recent public LLMs also permit the usage of alternative inputs such as images, either with or without a combination of text. We propose the usage of these image-based prompts for network configuration, where a source image could consist of the schematic of an existing or expected network topology. Along with a verbose text-based prompt, we theorize that the schematic input would serve as a data language that can reduce the ambiguity of an LLM for most network configuration scenarios.

While typical responses from an LLM mostly consist of natural language that is presented in readable format, it is essential to consider that the models are more than capable of providing responses that surpass the text-domain. A primary example of this capability is code generation, which has become a game changer in software development [14]. LLMs contain the ability to generate cohesive code from a wide variety of programming languages, and this skill is useful for the design of a response verification methodology [15]. When utilizing LLMs for network modification purposes, end-users must be concerned with the correctness of the configuration (low-level) commands that are generated, along with confidence on the overall high-level network capture for verification of network accuracy. For this reason, we propose to divide the verification methodology into two parts: the configuration commands of network equipment, such as routers, and the corresponding description of the network, along with a high-level image of a network topology, which can be visually analyzed.

In the low-level, the commands that are generated by the LLM are essential to end-users, further emphasizing the importance of a proper verification methodology to evaluate their effectiveness. Regardless of an input prompt's brevity, (public) LLMs are typically quite verbose in their responses, resulting in large amounts of information that must be processed. The generated response almost always contains a description of the network, or a summary of the configuration commands, followed by the commands themselves. As the description is quite arbitrary and often carries unnecessary information, we believe that identifying the configuration commands is the principal task in LLM response verification. When evaluating an individual command, we primarily consider the syntax, purpose, and impact of that command with respect to the problem statement. We then define a set of errors and distinct penalties, as shown in Table II, to ensure that each response is evaluated and errors are accounted for consistent penalties. The proposed methodology provides a numeric-based benchmark to evaluate a response generated by LLM.

TABLE II: Proposed Description-Evaluation Methodology.

Error	Penalty (%)
Invalid syntax	90
Invalid topology	75
Wrong number of network elements	60
Incorrect IP addressing	50
Incomplete or missing commands	40
Unnecessary commands	30

B. Schematic Prompting and Responding

As image-generation is not an explicit feature of text-based LLMs, and it is unknown whether LLMs are trained with image-based data, we propose the use of Scalable Vector Graphics (SVG) files, which are text-descriptive image (text) files. SVGs are primarily generated using a form of Extensible Markup Language (XML), and therefore can be constructed through the usage of code, which LLMs are quite capable of generating. We propose that any set of commands that are generated by the LLM can be modeled into an SVG-image to visualize the network topology. As such, we define a set of evaluation metrics in Table III for each SVG image that is generated to evaluate the accuracy of the network topology from a high-level perspective.

TABLE III: Proposed Image-Evaluation Methodology.

Error	Penalty (%)
Invalid syntax	100
Not a network	90
Invalid topology	75
Links not drawn between nodes	50
IP addresses not included	50
Labels are not drawn	45
Incorrect IP address (@)	20 / IP @
Incorrect drawn link	10 / Link

By integrating both prompt-engineering and response-assurance methodologies, we theorize that an LLM can be optimized such that it consistently provides highly accurate responses for a wide range of problem statements. In the next section, we evaluate the proposed approaches to achieve accurate LLM responses with a set of prompts to determine the accuracy of each model's performance.

IV. Evaluation

In our experiments we primarily evaluated OpenAI's state-of-the-art ChatGPT-4o model with various network topologies consisting of a varying number of network elements, and generated Cisco router commands to configure the network from scratch, along with SVG code depicting a network

topology to visualize the network design in an alternative representation. We used a wide variety of input prompts, in terms of provided values and number of networks, to cover network scenarios with different levels of complexity and specificity.

A. Experiment 1: An ideal prompt

It may be expected that an ideal prompt also causes the model to regurgitate information that it already knows, reducing the generation time that it has to intrinsically perform. It can be intuited that such a prompt would not typically be the preferred input an end-user would want to provide. However, it is important to understand how an LLM behaves in this ideal scenario to evaluate and establish an upper limit, such that a model can be bench-marked with alternative prompts. For this reason, we test an ideal prompt for a network scenario, and provide all information such as the number of network elements, IP addresses and required deliverables and in-turn ask the LLM to provide the Cisco router *commands* to configure the network routers, along with an SVG file to visualize the topology. We repeat this experiment 100 times and evaluate the accuracy of the provided description and SVG schematic, and document the overall performance of the model.

Figure 1 depicts the results obtained in each iteration in this experiment. The results primarily showcase a high description and SVG accuracy, with the description primarily hovering around 100% and the SVG image hovering around 90%. Although such outcomes would indeed be ideal, it must be noted that they lack congruency to each other, primarily due to the inherent randomness that stems from the LLM, which results in a semi-unique answer generated for each iteration. Such responses pose an additional challenge for a proper evaluation methodology, as the results of the current iteration could potentially be drastically different from that of the next one. The primary outliers in this experiment stem from the model either not generating all of the necessary commands or generating SVG images of invalid topologies, which typically points to a difficulty, not in understanding the prompt but rather evaluating it. Therefore, it can be inferred that while prompt-engineering methodologies can help alleviate the issue of intent recognition, they unfortunately cannot help with content generation due to the randomness that's inherent in the LLM. In the next few experiments, we evaluate multiple networking scenarios to further understand the impact of a typical input-prompt on the received response.

B. Experiment 2: Varied text-prompts based on scoring

In this experiment, we test three network configuration scenarios with five levels of input specificity to analyze the correctness of the LLM's response in each scenario. We begin with a prompt that contains none of the pitfalls described in Table I and slowly induce errors and re-calculate the score of the received commands and SVG across multiple runs. After twenty-five of such runs, we calculate the average accuracy of each scenario, and its input prompt grade to identify trends in the behavior of GPT-4o. This experiment aims to demonstrate both the variations that could potentially exist in an input prompt and the sensitivity of the LLM to such shifts.

Figure 2 shows the inherent sensitivity of the LLM with respect to an input-prompt, and the trends that GPT-4o responses with respect to various specificity levels. From the results, it can be inferred that a higher input prompt score typically results in greater overall accuracy regardless of the medium that is evaluated. However, because LLMs are primarily text-based models, it is no a surprise that the accuracy of a text description accuracy is marginally better than that of the generated SVG accuracy. Additionally, our results demonstrate that while an ideal prompt score of 100% is not always feasible, a subsequent score of 50% still provides a relatively helpful performance. Such behavior showcases the importance of utilizing a prompt with a well-defined input-score for a specified network configuration task, and demonstrates the inherent performance of an LLM with a wide range of erroneous information and ambiguity.

C. Experiment 3: Varied schematic prompts based on scoring

The idea of utilizing schematic prompts in conjunction with text is promising, as the approach has the potential to facilitate the identification of parameters used in a problem solution. In this approach, we start with a fully specified schematic, as described in Table III, and much like the previous experiment, we remove information to reduce the input prompt grade and re-run the evaluation multiple times. Along with the image, we utilize text to establish a clear set of deliverables and define the problem statement as a network design task, and alter the input schematic. We evaluate three different network scenarios, and run the schematic-prompts five times. We evaluate the description of the response, along with the router commands in conjunction with the generated SVG image.

Figure 3 shows a similar behavior as in the previous experiment. A higher input prompt grade results in a greater amount of information that is presented to the LLM which largely improves its chances at arriving at an optimal solution. Furthermore, when comparing with the results in Figure 2, it is noted that the accuracy with image-based prompts is marginally better than that of text-prompts for both of the evaluation methods. This boost to the accuracy is facilitated by the clear definition of expectations that an image can provide at a first-glance, as opposed to a medium like text in which we expect the model to perform most of the analysis.

The interesting trend of a lower SVG accuracy when compared with the description can be explained by a weakness in the LLM's image or code generation capabilities, which exist regardless of the type of input-prompt that is chosen. While intuitively it may seem that providing an image as an input can potentially increase the chances of receiving a proper SVG as an output, it must be noted that the process of generating an SVG remains constant. This further emphasizes the point that prompt-engineering techniques can be used to improve the performance of the LLM without directly altering its inherent composition. These results seem to indicate that while training and designing an LLM that is domain-specific is

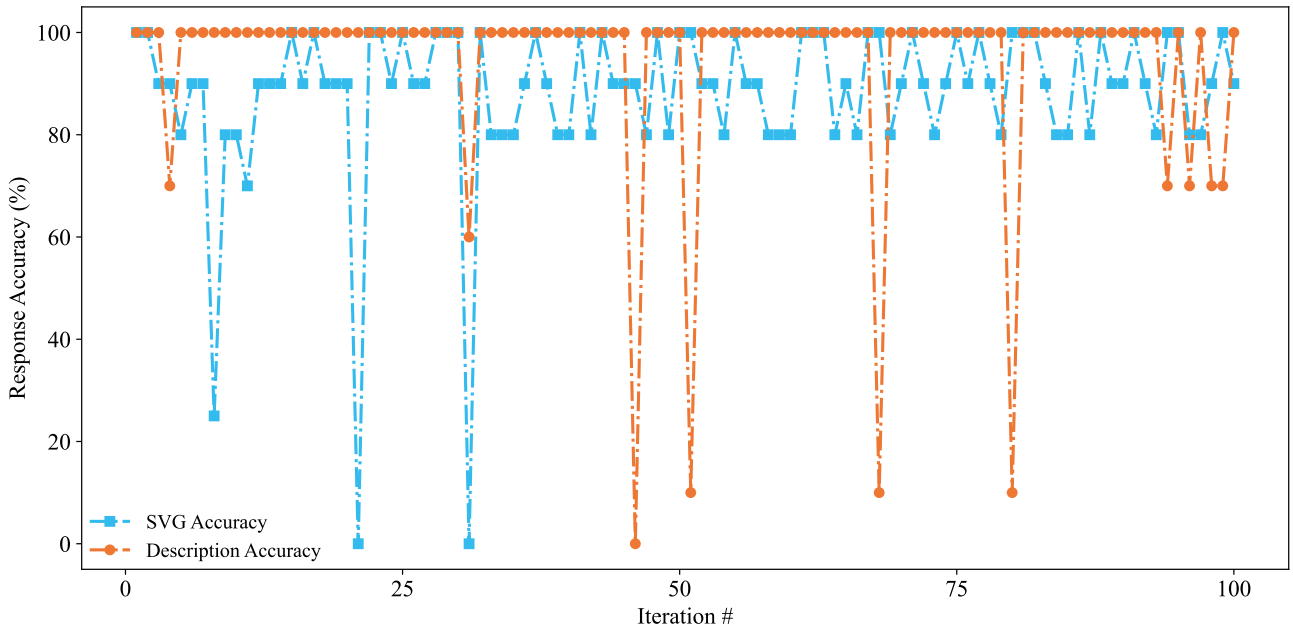

Fig. 1: Response accuracy of 100 iterations of an ideal prompt w/ GPT-4o.

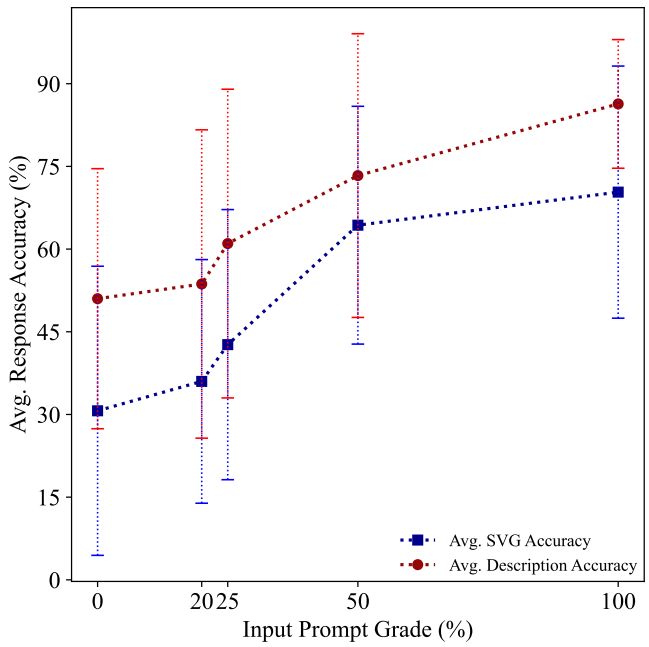

Fig. 2: Average response accuracy of text-prompts w/ GPT-4o.

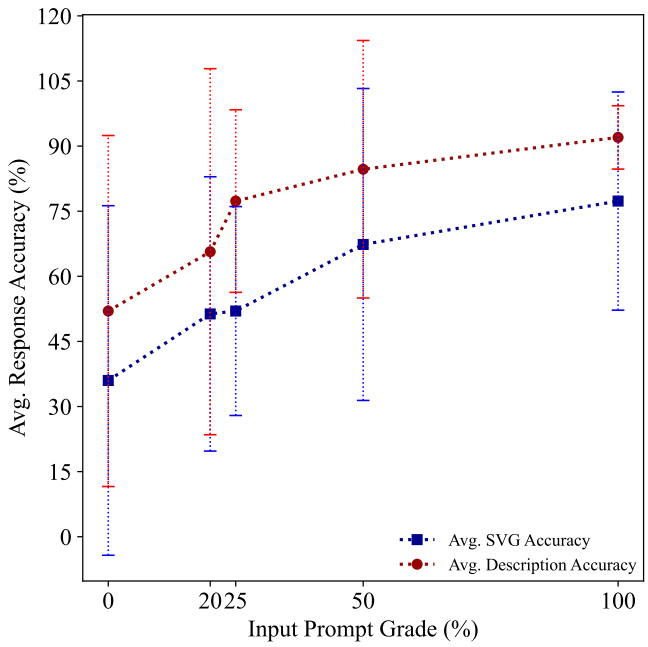

Fig. 3: Average response accuracy of image-prompts w/ GPT-4o.

beneficial, defining proper prompt-engineering methodologies are essential to achieve consistent performance.

D. Experiment 4: A comparison of state-of-the-art LLMs

As mentioned previously, there are a wide-variety of state-of-the-art general purpose LLMs that could potentially be used for network implementation purposes. In this experiment, we verify the performance of three such models: OpenAI's ChatGPT-4o, Google's Gemini and Microsoft's Copilot to determine the impact of the model with respect to the input prompt (Disclosure: this research received no support nor is associated with the holders of such LLMs). As such, we utilize a constant image input of a network topology and ask the respective model to generate the Cisco router commands to configure the network, along with an SVG to visualize it. We evaluate the performance of all three LLMs across five iterations and average the data to identify a clear trend across three different network configuration scenarios. The results,

shown in Figure 4, indicate that GPT-4o outperforms its competitors when it comes to network implementation scenarios. This advantage could stem from exposure to additional training performed on this LLM as opposed to the others, giving it the ability to consistently identify an optimal solution. GPT-4o often provided the same solution multiple times, which reduced the overall randomness and bode well for the overall performance.

It is equally interesting to see the trends that unfold between Copilot and Gemini, with the former outperforming the latter in SVG generation scenarios. Copilot's primary shortcoming when compared to GPT-4o was its lack of verbose definitions, which often lead to certain commands being either forgotten or missing entirely. With regards to SVG generation, the model often missed to both labeling network elements and indicating congruent IP addresses.

Unlike GPT and Copilot, Gemini's responses were often very short, providing a very brief or even no solution with respect to the description accuracy. In terms of the commands that were generated, some were often missed, similar to Copilot. A majority of Gemini's SVG drawings were also quite abstract, often depicting invalid or impossible networks entirely. We theorize that such unstable behavior could be attributed to the lack of overall training, that Gemini's competitors supposedly seem to have. Such behavior also indicates a need for domain-specific LLMs which could prove more useful for error-sensitive tasks such as network implementation.

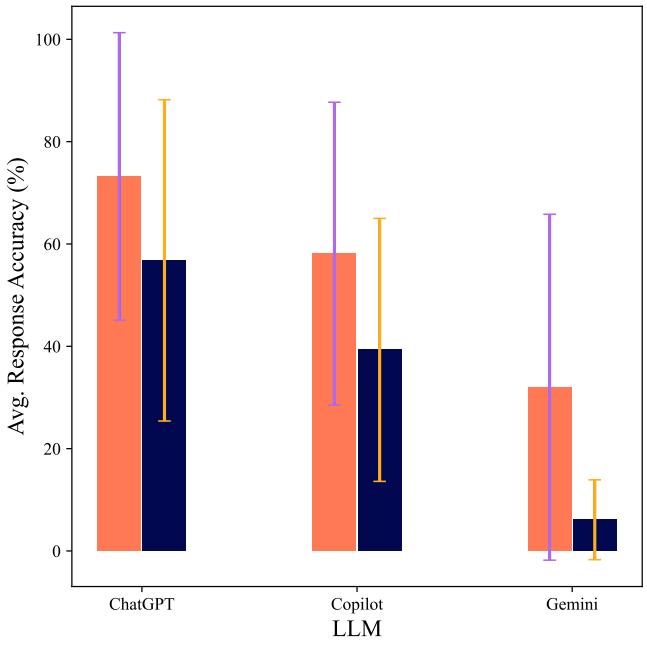

Fig. 4: Average response accuracy of image-prompts w/ GPT-4o.

V. Conclusions

Here, we proposed using a schematic (graphical) input prompt to improve LLM response accuracy and a scoring policy to evaluate LLM response in network design. We tested text and the proposed schematic prompts using various levels of specificity in various network scenarios and evaluated the response accuracy with the proposed scoring policy. We demonstrated that utilizing schematic prompting in conjunction with text increases the overall LLM performance. Our results strongly indicate that carefully designing the specifics of an input-prompt has the potential to benefit the response and inhibit the randomness of the model.

References

[1] Z. Ben Houidi and D. Rossi, "Neural language models for network configuration: Opportunities and reality check," *Computer Communications*, vol. 193, p. 118–125, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2022.06.035

[2] A. Çelen, G. Han, K. Schindler, L. V. Gool, I. Armeni, A. Obukhov, and X. Wang, "I-Design: Personalized LLM Interior Designer," 2024. [Online]. Available: https://arxiv.org/abs/2404.02838

[3] S. Wu, A. Khasahmadi, M. Katz, P. K. Jayaraman, Y. Pu, K. Willis, and B. Liu, "CAD-LLM: Large Language Model for CAD Generation," in *Proceedings of the neural information processing systems conference. neurIPS*, 2023.

[4] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu, X. Liu, C. Zhang, X. Wang, and J. Liu, "Large Language Model (LLM) for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities," 2024. [Online]. Available: https://arxiv.org/abs/2405.10825

[5] R. Rojas-Cessa, *Interconnections for Computer Communications and Packet Networks*. CRC Press, 2016.

[6] V. Komanduri, C. Wang, and R. Rojas-Cessa, "Comparing link sharing and flow completion time in traditional and learning-based tcp," in *2024 IEEE 25th International Conference on High Performance Switching and Routing (HPSR)*, 2024, pp. 167–172.

[7] R. Mondal, A. Tang, R. Beckett, T. Millstein, and G. Varghese, "What do LLMs need to Synthesize Correct Router Configurations?" 2023. [Online]. Available: https://arxiv.org/abs/2307.04945

[8] M. Besta, L. Paleari, A. Kubicek, P. Nyczyk, R. Gerstenberger, P. Iff, T. Lehmann, H. Niewiadomski, and T. Hoefler, "CheckEmbed: Effective Verification of LLM Solutions to Open-Ended Tasks," 2024. [Online]. Available: https://arxiv.org/abs/2406.02524

[9] G. Sun, Y. Wang, D. Niyato, J. Wang, X. Wang, H. V. Poor, and K. B. Letaief, "Large Language Model (LLM)-enabled Graphs in Dynamic Networking," 2024. [Online]. Available: https://arxiv.org/abs/2407.20840

[10] M.-V. Dumitru, V.-A. Bădoiu, A. M. Gherghescu, and C. Raiciu, "Generating P4 Dataplanes Using LLMs," in *2024 IEEE 25th International Conference on High Performance Switching and Routing (HPSR)*, 2024, pp. 31–36.

[11] O. G. Lira, O. M. Caicedo, and N. L. S. da Fonseca, "Large Language Models for Zero Touch Network Configuration Management," 2024. [Online]. Available: https://arxiv.org/abs/2408.13298

[12] D. Wu, X. Wang, Y. Qiao, Z. Wang, J. Jiang, S. Cui, and F. Wang, "NetLLM: Adapting Large Language Models for Networking," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. ACM SIGCOMM '24. ACM, Aug. 2024, p. 661–678. [Online]. Available: http://dx.doi.org/10.1145/3651890.3672268

[13] B. Ifland, E. Duani, R. Krief, M. Ohana, A. Zilberman, A. Murillo, O. Manor, O. Lavi, H. Kenji, A. Shabtai, Y. Elovici, and R. Puzis, "GeNet: A Multimodal LLM-Based Co-Pilot for Network Topology and Configuration," 2024. [Online]. Available: https://arxiv.org/abs/2407.08249

[14] F. Mu, L. Shi, S. Wang, Z. Yu, B. Zhang, C. Wang, S. Liu, and Q. Wang, "ClarifyGPT: A Framework for Enhancing LLM-Based Code Generation via Requirements Clarification," *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3660810

[15] O. Chatterjee, P. Aggarwal, S. Samanta, T. Dai, P. Mohapatra, D. Kar, R. Mahindru, S. Barbieri, E. Postea, B. Blancett, and A. D. Magalhaes, "ScriptSmith: A Unified LLM Framework for Enhancing IT Operations via Automated Bash Script Generation, Assessment, and Refinement," 2024. [Online]. Available: https://arxiv.org/abs/2409.17166