Prompting the Future: Integrating Generative LLMs and Requirements Engineering

Andreas Vogelsang[1]

[1]University of Cologne, Germany

Abstract

This paper provides an overview of a keynote presentation given at the 7th Workshop on Natural Language Processing for Requirements Engineering (NLP4RE) regarding the utilization of generative Large Language Models (LLMs) for addressing Requirements Engineering (RE) tasks. It highlights the transformative impact of decoder-only LLMs, exemplified by models like GPT, on various domains, including RE, owing to their remarkable language understanding and generation capabilities. The discussion centers on how decoder-only LLMs can revolutionize requirements elicitation, specification, and validation processes, potentially reshaping the RE landscape. The paper is structured into two main sections: the first explores the application of decoder-only models in automating RE tasks, emphasizing richer output and novel interaction paradigms, while the second segment emphasizes the pivotal role of precise requirements in crafting effective prompts for interacting with these models, drawing parallels between requirements specification techniques and prompt engineering strategies.

Keywords

Requirements Engineering, Natural Language Processing, Large Language Models, Generative AI

1. Introduction

As part of the 7th Workshop on Natural Language Processing for Requirements Engineering (NLP4RE), I gave a keynote on the use of generative LLMs for solving Requirements Engineering (RE) tasks. This paper summarizes the main points of the keynote.

Decoder-only LLMs, such as GPT, have revolutionized how we interact with artificial intelligence. Their ability to understand, generate, and manipulate language presents unprecedented opportunities and challenges across various disciplines, including RE. Decoder-only LLMs have the potential to redefine the landscape of requirements elicitation, specification, and validation.

The article is structured into two primary segments. The first part delves into the application of decoder-only models in automating RE tasks. It explores how these models can assist in accurately capturing and specifying requirements, generating requirement documents, and automating the verification of requirements consistency and completeness. By examining case studies and current research, this section will highlight the transformative potential of

In: D. Mendez, A. Moreira, J. Horkoff, T. Weyer, M. Daneva, M. Unterkalmsteiner, S. Bühne, J. Hehn, B. Penzenstadler, N. Condori-Fernández, O. Dieste, R. Guizzardi, K. M. Habibullah, A. Perini, A. Susi, S. Abualhaija, C. Arora, D. Dell'Anna, A. Ferrari, S. Ghanavati, F. Dalpiaz, J. Steghöfer, A. Rachmann, J. Gulden, A. Müller, M. Beck, D. Birkmeier, A. Herrmann, P. Mennig, K. Schneider. Joint Proceedings of REFSQ-2024 Workshops, Doctoral Symposium, Posters & Tools Track, and Education and Training Track. Co-located with REFSQ 2024. Winterthur, Switzerland, April 8, 2024.

vogelsang@cs.uni-koeln.de (A. Vogelsang)
0000-0003-1041-0815 (A. Vogelsang)

decoder-only LLMs in enhancing efficiency, accuracy, and comprehensiveness in RE.

The second segment of the paper positions RE as a critical discipline for developing well-crafted prompts essential for interacting with decoder-only LLMs. It underscores the importance of precise, unambiguous, and comprehensive requirements in formulating prompts that elicit accurate and relevant responses from the models. This part will also discuss the art and science of crafting effective prompts, drawing parallels between requirements specification techniques and prompt engineering strategies.

2. Preliminaries: Decoder-only (Generative) LLMs

Decoder-only LLMs have been designed to generate text. To support the generative capabilities of decoder-only LLMs, they are primarily pre-trained with a next-word prediction (NWP) objective, where the models predict the next word or words in a given sequence of words. After pre-training, decoder-only LLMs are triggered by a so-called *prompt*. A prompt is a textual input instructing the generative LLM to generate the desired response. Feeding decoder-only LLMs with prompts offers a new paradigm for interaction. In contrast to non-generative LLMs, it is not necessary to encode information about the task and the input in a smart way. Instead, the task and the input can be expressed in natural language and passed directly to the model. The model's output is also richer than for non-generative LLMs since it produces (customizable) text instead of confidence values for predefined outcomes. A prompt contains any of the following elements:

- **Instruction:** a specific task or instruction you want the model to perform
- **Context:** external information or additional context that can steer the model to better responses
- **Input Data:** the input or question that we are interested in finding a response for
- **Output Indicator:** the type or format of the output.

You do not need all four elements for a prompt, and the format depends on the task at hand. Although prompting an LLM sounds relatively straightforward, the creation and exact phrasing of a prompt is crucial for the quality of the LLM output. For most tasks, it is necessary to experiment with different prompts and iteratively refine them to yield the best results. This so-called *prompt engineering* step is similar to feature engineering in more traditional ML approaches. Prompt engineering is a relatively new field where only heuristics and simple rules exist that help create effective prompts.

3. Using Generative LLMs for RE tasks

Generative LLMs provide a new way to solve RE tasks. Traditional predictive LLMs are trained or fine-tuned to predict the most likely output for a given input. Usually, the set of possible outputs is constrained. It is either a set of classes [1, 2], a subset of the input (i.e., information retrieval [3]), or a numeric value in a defined range (e.g., similarity estimation).

With generative LLMs, a specific RE task is written as a prompt, which serves as input. The prompt usually contains the task and the input data to be processed. The model's output provides the answer to the specific task.

Here is an example of how a trace link recovery task can be formulated as prompt [4]:

User Prompt:

Below are artifacts from the same software system. Is there a traceability link between (1) and (2)?

(1) The DPU-TMALI shall utilize SCM_DCI_SR, along with ERRNO provided by DPU-DCI to decode errors and place them on an error queue for DPU-CCM.

(2) Error Collection and Reporting The ccmErrEnq() function tracks the last error reported and its frequency of occurrence. Once an error code has been reported it becomes the previously reported error code maintained by ccmErrEnq(). A repetition count is then incremented for each subsequent, consecutively reported, identical instance of this previously reported error. If this error code is reported more than once in one high-rate housekeeping reporting period, then a special error, S_ccm_ERR_REPEAT is enqueued with the repetition count for the error encoded in the least significant byte. This mechanism effectively reduces the potential for housekeeping telemetry to become flooded with a single repeated error.

Assistant:

Yes, there is a traceability link between (1) and (2). (1) mentions that the DPU-TMALI places errors on an error queue for the DPU-CCM.

(2) describes the error collection and reporting mechanism of the DPU-CCM, specifically referring to enqueuing errors.

So (2) provides an implementation detail for the error queuing functionality briefly mentioned in (1). Therefore, there is a traceability link between these two artifacts in the software system.

The quality of the provided output depends on the specific prompt. Prompts can be refined by refining any of their components (i.e., instructions, context, input question).

Providing more context and instructions is an effective strategy to increase the semantic quality of the output. In a recent publication, we describe *few-shot prompting*, *instruction fine-tuning*, and *retrieval augmented generation*, which can be used to refine the prompt and increase the model's performance [5].

Besides the novel input paradigm, LLMs also offer two novel ways to work with the output.

We can ask generative LLMs to augment the response with rationales and explanations. In contrast to predictive LLMs, which only provide the prediction as output[1], generative LLMs can be instructed to provide explanations along the result. As also shown in the example above, the assistant provides the answer ("yes, there is a trace link") followed by an explanation. This additional information impacts the perception of the human analyst using the tool. It may also increase the chances that humans can uncover mispredictions by the model and also help

[1]confidence values may be considered as output augmentation

humans to improve quality flaws.

We can enter into a dialog with a generative LLM. The second novel advantage is that generative LLMs can incorporate prior answers as context into new prompts. This gives the user the possibility to ask for additional information or even correct certain details of an answer. The generative LLM can respond to this additional human input to refine, clarify, or even correct its answer.

If, for example, we have spotted a mistake in the answer, we may correct the model and get a refined answer:

User Prompt:

The explanations is not correct. (2) does not describe the DPU-CCM but another component.

Assistant:

I apologize for the confusion in my previous response. Based on the information provided, there is no direct traceability link between (1) and (2) in terms of specific components or functionalities. (1) describes the error handling process in the DPU-TMALI component, while (2) describes the error collection and reporting mechanism in a different component that is not explicitly mentioned.

Without further context or explicit connections between the components mentioned in (1) and (2), it is not possible to establish a direct traceability link between the two artifacts.

4. Using RE for Effective Prompting

In the rapidly advancing landscape of artificial intelligence (AI), prompt engineering emerged as a new discipline concerning the development and optimization of prompts for large language models (LLMs). Various prompting techniques have been introduced by different sources, ranging from blog articles to reputable publications. However, the vast majority seem to lack a scientific foundation and are more based on trial and error. Some of these rules are:

- **Start simple:** start with a simple prompt and build on it
- **Call to Action:** start the prompt with an action word like "Write", "Create", or "Summarize" instead of "Can you"
- **Add Context:** add specific and relevant context to the task you want to perform
- **Add Expectations:** add clear and direct expectations for the content, like how long it should be and what to include

White et al. [6] describe a catalog of prompt engineering techniques presented in pattern form that have been applied to solve common problems when conversing with LLMs. Prompt patterns are a knowledge transfer method analogous to software patterns since they provide reusable solutions to common problems faced in a particular context, i.e., output generation and interaction when working with LLMs.

Arora et al. [7] present an approach that automatically converts task inputs to effective prompt structures and uses weak supervision, a method for aggregating signals from noisy labelers, to aggregate the responses.

Zhou et al. [8] present the Automatic Prompt Engineer, which automatically generates instructions for a task that is specified via output demonstrations: it generates several instruction candidates, either via direct inference or a recursive process based on semantic similarity, executes them using the target model, and selects the most appropriate instruction based on computed evaluation scores.

Requirements Engineering research has a long tradition of thinking about and coming up with methods to describe stakeholder wishes in a precise manner.

Recently, we investigated reproducible indicators within prompts that may predict a loss of quality or flaw. We base these flaw indicators on established requirements smells, which are reliable indicators for requirements quality [9, 10]. Our initial experiments showed interesting results. For example, in a code generation task, ChatGPT interpreted the vague requirement "If the points of one player are too low, the game ends and the player loses the game." by setting the boundary to 0 points without asking for clarification or hinting at the ambiguity:

5. Conclusions and Open Questions

Generative LLMs offer new ways to support requirements analysts in performing their tasks. In contrast to predictive LLMs, generative LLMs can be instructed to augment their output with explanatory information. Users may refine or correct the answer by entering into a dialog with the generative LLM. These novel possibilities also come with challenges for RE researchers. In particular, the capabilities of LLMs must be evaluated differently. This starts with different metrics that are used to compare generated text (e.g., BLEU and ROUGE instead of precision and recall). Additionally, we need new ways to assess the effectiveness of an approach when evaluated with humans in the loop.

Another stream of research is necessary to incorporate RE knowledge in the field of prompt engineering. This includes guidance when creating precise prompts but also more complex tasks like how to break down more abstract goals into smaller subgoals that may finally be translated into single prompts.

Acknowledgments

This paper and the corresponding keynote are based on insights from several joint publications and discussions with esteemed colleagues, especially, Jannik Fischbach, Alessio Ferrari, Fabiano Dalpiaz, Chetan Arora, Sallam Abualhaija, and Gouri Deshpande.

References

[1] J. Winkler, A. Vogelsang, Automatic classification of requirements based on convolutional neural networks, in: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), IEEE, 2016, pp. 39–45.

[2] M. Binder, A. Vogt, A. Bajraktari, A. Vogelsang, Automatically Classifying Kano Model Factors in App Reviews, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer Nature Switzerland Cham, 2023, pp. 245–261.

[3] J. Fischbach, B. Hauptmann, L. Konwitschny, D. Spies, A. Vogelsang, Towards causality extraction from requirements, in: 2020 IEEE 28th International Requirements Engineering Conference (RE), IEEE, 2020, pp. 388–393.

[4] A. D. Rodriguez, K. R. Dearstyne, J. Cleland-Huang, Prompts matter: Insights and strategies for prompt engineering in automated software traceability, in: IEEE 31st International Requirements Engineering Conference Workshops (REW), 2023, pp. 455–464. doi:10.1109/REW57809.2023.00087.

[5] A. Vogelsang, J. Fischbach, Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline, 2024. doi:10.48550/ARXIV.2402.13823.

[6] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, A prompt pattern catalog to enhance prompt engineering with ChatGPT, 2023. doi:10.48550/ARXIV.2302.11382.

[7] S. Arora, A. Narayan, M. F. Chen, L. Orr, N. Guha, K. Bhatia, I. Chami, F. Sala, C. Ré, Ask me anything: A simple strategy for prompting language models, 2022. doi:10.48550/ARXIV.2210.02441.

[8] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, J. Ba, Large language models are human-level prompt engineers, 2022. doi:10.48550/ARXIV.2211.01910.

[9] H. Femmer, D. Méndez Fernández, S. Wagner, S. Eder, Rapid quality assurance with requirements smells, Journal of Systems and Software 123 (2017) 190–213. doi:10.1016/j.jss.2016.02.047.

[10] J. Frattini, L. Montgomery, J. Fischbach, M. Unterkalmsteiner, D. Mendez, D. Fucci, A live extensible ontology of quality factors for textual requirements, in: IEEE 30th International Requirements Engineering Conference (RE), IEEE, 2022. doi:10.1109/re54965.2022.00041.