North American Journal of Engineering and Research

Est. 2020

AutoPrecisePrompts: Automated LLM-based Prompt Engineering for Structured Data Processing

Praneeth Vadlapati
Email: praneethv@arizona.edu

Abstract

Processing and manipulating structured data with Language Models has become vital for various use cases. However, not all language models may follow the expected output consistently, necessitating reattempts until the requirements are met. Repeated queries increase the resources and time required to process the information. Such problems necessitate effective Prompt Engineering and testing across various test cases. Prompt Engineering, when performed without automation, requires a larger workforce and significant time and resources. An alternative approach, such as Prompt Tuning, introduces further challenges. To solve all the challenges, this research proposes an AI-driven automated prompt optimization system designed to enhance the accuracy of prompts for various AI applications, using minimal time and resources. By iteratively testing prompts using a smaller language model and adjusting the prompt with the help of a Large Language Model until optimal performance is achieved, the system automates the process of optimizing prompts. Without requiring a training process before optimization, this approach ensures the reusability and transparency of optimized prompts to use across different language models. The system uses the expected output to offer a way for organizations to overcome the difficulties associated with manual-only prompt engineering. The system offers a solution to create concise, high-quality prompts that yield the desired accuracy. During the experiment, the system achieved the expected accuracy using only two iterations. The prompt led to satisfactory accuracy using multiple Language Models, proving the reusability of the prompt.

Keywords: Artificial Intelligence (AI), Large Language Models, Prompt Engineering, Prompt Optimization, Prompting, Model Performance, Model Reliability, Structured Data, AI Transparency

Introduction

Structured data is essential to numerous organizations. Language Models (LMs) have a large knowledge base. Processing structured data in formats such as CSV and JSON using LMs requires optimal prompts to ensure the model responds in the expected structure. “Prompting is programming.” [1] Making models respond by following our requirements involves challenges [2]. Language models were often found to miss some parts or add unnecessary parts that are not required. Overcoming such challenges involves multiple methods, including Prompt Engineering, Prompt Tuning, and Fine-tuning an LM. More methods include querying repeatedly until the model responds in the expected format and using a larger model by expecting a response in the expected structure. Repeated queries consume more time and resources.

Challenges with Other Approaches

Manual-only prompt engineering requires significant time, resources, and workforce. Numerous iterations are required to attain an acceptable accuracy. Human experts often miss crafting the optimal prompt or lack prompting skills [2]. The approach of Prompt Tuning [3] involves challenges and may fail to achieve the same results as Prompt Engineering. It relies on the performance of an optimization algorithm. It involves an additional step of tuning the prompt parameters.

Soft prompting is flexible as it involves embeddings instead of fixed prompts. However, it can be challenging to interpret and control, as the internal representations are not transparent and hard to trust directly [4]. Fine-tuning the model consumes time and resources. Fine-tuning does not always ensure satisfactory accuracy for the requirements [5]. In the current pace of new models getting released, switching to a newer model is not flexible in the case of fine-tuning. These methods need special algorithms or training datasets, which have the potential to reduce their generalizability to new or unseen tasks.

Automated Prompt Optimization and Its Benefits

This paper proposes a system for automated optimization of prompts, which ensures transparency in the prompts and the reusability of the prompts across multiple models. This approach does not involve a training process and ensures the same advantages as alternative approaches without the same disadvantages. Prompts have interpretability and can be understood by anyone who is not an expert in developing LMs.

Advantages of Small Language Models

When compared to larger models, smaller models are faster and consume lesser resources [6], which allows them to process a large amount of data in less time. Smaller models are being developed rapidly. When using smaller models, prompts should be optimized to ensure reliability.

Literature Review

MAPO [7] and Robust Prompt Optimization [8] have improved model-specific prompt tuning and robustness against distribution shifts, but they rely on task-specific data and manual intervention. Even though approaches such as APE [9] and AutoHint [10] automate some aspects of prompt engineering, their focus is not on processing structured data and requires datasets for training. Existing research on prompt optimization focused on adaptability and robustness and left a gap in the effectiveness and scalability of handling structured data using language models while ensuring the reusability of the responses and interpretability of prompts.

Despite the progress of advancements, gaps exist in interaction with unseen scenarios, scalability, robustness, and adaptation to task complexity. Addressing such gaps remains pivotal for further progress in Prompt Optimization for LMs. Existing research focuses on automated prompt tuning and soft prompting, leaving a research gap in automated Prompt Engineering to make language models process structured data efficiently, unlike processing unstructured data. Existing work uses datasets for training. However, the system proposed in this paper uses only an existing prompt and the expected output. The system does not undergo a training process, which consumes extra resources, time, and additional workforce for various organizations such as startups and non-profits.

Methods

Testing the Initial Prompt using Expected Output

GPT-4 Turbo [11] is the Large Language Model selected to rewrite the prompts. Claude Instant 1.2 [12] is the smaller model selected for affordable usage to test the prompts. An initial prompt template, as well as the expected response, are written for the experiment. Ten trials are conducted using the small LM to check the accuracy of the initial prompt.

Prompt Template 1. **Initial Prompt Template from the User**

~~~
Here is input data: {structured_input_data}.
Provide the name and age of people whose age is below 35.
~~~

Sample Value 1. **Structured Sample CSV Data Provided as Input**

~~~
```csv
Name,Gender,Age,City
John,Male,25,NYC
Jane,Female,30,LA
Doe,Male,38,Chicago
Emily,Female,48,Houston
Henry,Male,66,Philadelphia
```
~~~

Sample Value 2. **Expected Structured CSV Response**

~~~
```csv
Name,Age
John,25
Jane,30
```
~~~

Sample Value 3. **Structured Sample JSON Data Provided as Input**

~~~
```json
[
   {"Name": "John", "Gender": "Male",
      "Age": 25, "City": "NYC"},

   {"Name": "Jane", "Gender": "Female",
      "Age": 30, "City": "LA"},

   {"Name": "Doe", "Gender": "Male",
      "Age": 38, "City": "Chicago"},

   {"Name": "Emily", "Gender": "Female",
      "Age": 48, "City": "Houston"},

   {"Name": "Henry", "Gender": "Male",
      "Age": 66, "City": "Philadelphia"}
]
```
~~~

Sample Value 4. **Expected Structured JSON Response**

~~~
```json
[
   {"Name": "John", "Age": 25},
   {"Name": "Jane", "Age": 30}
]
```
~~~

Optimizing Prompts using the Larger Model

A prompt is assumed as optimal if it leads to at least 90% accuracy. The larger model is used to evaluate the prompt and response against the expected response to attempt to optimize the prompt template. The accuracy of the smaller model is checked again using the new prompt. The prompt template is regenerated up to five times until an optimal prompt template has been generated.

Prompt Template 2. **Prompt to Optimize the Current Prompt**

```
Value of structured_input_data variable (includes backticks):
{structured_input_data}
 ---
Processing criteria (from Initial Prompt):
{processing_criteria}
 ---
Current prompt template: {current_prompt_template}
 ---
Accuracy of current prompt: {current_accuracy}
Current response: {current_response}
 ---
Expected response: {expected_response}
 ---

Act as a Prompt Engineer and an expert Linguist. Write prompt to process structured data using language model. Rewrite prompt template to generate expected response (including its special characters and backticks). Use example_response placeholder to indicate a sample response. Don't add your own sample in the template.

Don't include answer or expected response. CSV responses must include expected column names without extra columns. Write only new prompt template without any other text. At last, emphasize on the processing criteria I mentioned. Backticks and format must be exactly same as the example response.

Avoid backticks like ```. Mention "{structured_input_data}" placeholder to indicate input data. Make sure response includes backticks. Don't miss both placeholders mentioned in curly braces. Add "**Write like**: {example_response}" as placeholder to mention example response.
```

Shortening the Prompt using the Larger Model

After constant improvement, the final prompt would be much longer than the initial prompt. Hence, concise prompts are generated. If the accuracy produced by the shorter prompt is the same as the optimized prompt, it is considered to be used.

Prompt Template 3. **Prompt to Shorten the Current Prompt**

```
Current prompt template:
{current_prompt_template}
---
Be a Prompt Engineer. Shorten the above prompt template. Make sure the prompt is short and concise. Retain the placeholders values and key information. Do not remove placeholders 'structured_input_data' and 'example_response'. Return only the shortened prompt without any other text. When a language model uses the prompt to generate response, backticks and format must be exactly same as the example response. Make sure response from the model includes the formatted data inside backticks with the format like
{current_response}
```

Testing the Prompt using Multiple Language Models

Final testing is performed on multiple models to explore the reusability of the prompts generated by the system. Claude Instant 1.2 [12], GPT 3.5 Turbo [13], and Llama 2 (13B) [14] are the smaller LMs that are selected. GPT-4 Turbo [11] is a large model selected to experiment on whether using a large model would result in accurate structured responses.

Results

Generating Prompts using the Larger Model

Optimal prompts were generated successfully by the large model using only two iterations of optimizing prompts in both CSV and JSON test cases. Shortening the prompt has led to a reduction in the accuracy for five attempts for both test cases.

Prompt Template 4. **Optimized Prompt for CSV test case**

```
Please filter the data contained within
{structured_input_data} to identify individuals who are younger than 35 years old and provide a CSV formatted list that includes only their Name and Age. Ensure the output is presented with the exact column headers as in the input data and that it is enclosed within backticks, consistent with
{example_response}.
Apply special attention to meet the specified processing criteria and maintain the integrity of the CSV format, including leading and enclosing backticks.

**Write like**: {example_response}
```

Prompt Template 5. **Optimized Prompt for JSON test case**

```
Given the following data in JSON format:
{structured_input_data}, extract and provide the name and age of individuals who are under the age of 35 in the same JSON format. Ensure that your output matches the precise structure depicted here:
**Write like**: {example_response}
```

Testing the Prompt using Multiple Language Models

The accuracy has been tested on multiple models using the same prompt to test the reusability. Common mistakes included backticks (```) not found as per the example response to indicate the structured data and the format "CSV" or "JSON" not being mentioned by the model in the response. Rarely did the models return Python code to process the data, or extra rows were added without removal based on the criteria. For the JSON test case, Claude displayed satisfactory accuracy with the initial prompt, and hence, GPT-3.5 Turbo was used as the base small model. The prompt optimized for Claude has been optimal for Llama 2 (with an accuracy of 80%) even though the prompt was not optimized for Llama 2. The prompt can be further engineered to make other models respond in the expected format.

TABLE I. ACCURACY USING EACH PROMPT WITH MULTIPLE MODELS IN CSV TEST CASE

Model	**Accuracy using each prompt**	
	Initial Prompt	**Optimized Prompt**
Claude Instant	0%	**90%**
GPT-3.5 Turbo	40%	**90%**
GPT-4 Turbo	10%	**100%**
Llama 2	0%	0%

TABLE II. ACCURACY USING EACH PROMPT WITH MULTIPLE MODELS IN JSON TEST CASE

Model	**Accuracy using each prompt**	
	Initial Prompt	**Optimized Prompt**
GPT-3.5 Turbo	0%	**100%**
Claude Instant	90%	**100%**
GPT-4 Turbo	0%	**100%**
Llama 2	0%	**80%**

Discussion and Limitations

The purpose of this study is only to experiment with an iterative approach to automate the Prompt Engineering process. The system has the possibility to make a difference in the efficiency of processing structured data for various organizations. This study involves the usage of only text and does not involve multimodal data such as images. Standard benchmarks can be used to test the system and compare it against base models. The system has not been experimented with to handle "impossible responses," such as harmful responses, which most of the language models refuse to generate. This research does not involve handling when the input, prompt, or expected response has more tokens than what the model is designed to process. The system proposed in this paper is used to evaluate a new approach of prompt engineering to process structured data. However, it is not tested using a wide variety of test cases and data formats other than CSV and JSON.

Conclusion

In order to address the inefficiencies of manual-only prompt engineering and the limitations of alternative approaches like fine-tuning and prompt tuning, this paper presents an AI-driven method to optimize prompts. Using a larger model did not ensure a satisfactory accuracy of the response structure, and automated prompt engineering has been beneficial. The system proposed in this paper has generated successful prompts to process structured data using language models. The system has successfully automated the process of improving prompts using only two iterations, ensuring satisfactory accuracy without necessitating large training datasets or the consumption of huge amounts of resources. It was proved that the final optimized prompts could be reused with other language models in both CSV and JSON test cases, proving the reusability of the prompt. The system failed in the attempts to use a language model to further shorten the prompts to achieve a balance between prompt conciseness and response accuracy. Future work might include testing the system with a broader range of response structures, diverse test cases, and numerous language models, further enhancing the robustness and scalability of the automated optimization of prompts.

References

[1] L. Beurer-Kellner, M. Fischer, and M. Vechev, "Prompting Is Programming: A Query Language for Large Language Models," Proc. ACM Program. Lang., vol. 7, no. PLDI, Jun. 2023, doi: 10.1145/3591300.

[2] J. D. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts," in Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, in CHI '23. New York, NY, USA: Association for Computing Machinery, Apr. 2023. doi: 10.1145/3544548.3581388.

[3] Y. Wang, J. Chauhan, W. Wang, and C.-J. Hsieh, "Universality and Limitations of Prompt Tuning," in Advances in Neural Information Processing Systems, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., Curran Associates, Inc., 2023, pp. 75623–75643. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/eef6aecfe050b556c6a48d9c16b15558-Paper-Conference.pdf

[4] L. Bailey, G. Ahdritz, A. Kleiman, S. Swaroop, F. Doshi-Velez, and W. Pan, "Soft prompting might be a bug, not a feature," in ICML 2023 Challenges of Deploying Generative AI Workshop, Jun. 2023.

[5] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha, "Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs," Dec. 2023, arXiv:2312.05934. [Online]. Available: https://arxiv.org/abs/2312.05934

[6] Q. Fournier, G. M. Caron, and D. Aloise, "A Practical Survey on Faster and Lighter Transformers," ACM Comput. Surv., vol. 55, no. 14s, Jul. 2023, doi: 10.1145/3586074.

[7] Y. Chen et al., "MAPO: Boosting Large Language Model Performance with Model-Adaptive Prompt Optimization,"

in Findings of the Association for Computational Linguistics: EMNLP 2023, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 3279–3304. doi: 10.18653/v1/2023.findings-emnlp.215.

[8] M. Li, W. Wang, F. Feng, Y. Cao, J. Zhang, and T.-S. Chua, “Robust Prompt Optimization for Large Language Models Against Distribution Shifts,” in Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 1539–1554. doi: 10.18653/v1/2023.emnlp-main.95.

[9] Y. Zhou et al., “Large Language Models are Human-Level Prompt Engineers,” in The Eleventh International Conference on Learning Representations, Feb. 2023. [Online]. Available: https://openreview.net/forum?id=92gvk82DE-

[10] H. Sun et al., “AutoHint: Automatic Prompt Optimization with Hint Generation,” Aug. 2023, arXiv:2307.07415. [Online]. Available: https://arxiv.org/abs/2307.07415

[11] OpenAI, “GPT-4 Turbo (gpt-4-1106-preview) [Large Language Model].” [Online]. Available: https://openai.com/index/new-models-and-developer-products-announced-at-devday/

[12] Anthropic, “Claude Instant (1.2) [Language Model].” [Online]. Available: https://www.anthropic.com/news/releasing-claude-instant-1-2

[13] OpenAI, “GPT-3.5 Turbo (gpt-3.5-turbo-1106) [Large Language Model].” [Online]. Available: https://openai.com/index/new-models-and-developer-products-announced-at-devday/

[14] Meta, “Llama 2 (13B) [Language Model].” [Online]. Available: https://github.com/meta-llama/llama/blob/main/MODEL_CARD.md