Enhanced Calendar Assistant through Prompt Engineering

Zachary Decker

5 May 2024

Abstract

Summary: This project is a continuation of my final project from last fall in my Deep Learning class. The goal was to enhance a barebones prototype of an LLM calendar assistant by improving the prototype and leveraging advanced prompt engineering techniques. The project involved creating a new prototype using React and Node.js, followed by experimenting with various prompting techniques. The evaluation showed improvements in handling both simple and complex calendar modifications.

Keywords: Calendar Assistant, LLM, Prompt Engineering, React, Node.js, ChatGPT

1 Introduction

1.1 Problem Statement

Recently LLM's have been shown to be capable of a variety of tasks including ones that require complex reasoning. To experiment with the capabilities of modern LLMs as a final project of a deep learning class in the fall of 2023, a prototype of a LLM-powered calendar assistant was created intended to be capable of making modifications to a users calendar based on various queries. Additionally various prompt engineering techniques were leveraged to improve the output. The goal of this project was to expand upon the original by performing a larger and more structured analysis of the effectiveness of different prompting techniques while improving quality of outputs and usability of the web interface.

1.2 Literature Review

With regards to finding prompting techniques to examine for this project, two main survey papers were used. The first titled ”A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT” was produced by Vanderbilt university in 2022 (1). This paper not only acted as a survey of prominent techniques but also proposed a framework for organizing and studying prompting techniques. This framework was meant to emulate software design patterns. Where different software design patters could be categorized by the type of problem they solved, the proposed ”prompt patterns” are categorized by the kind of task one is assigning to the LLM. From this paper, Meta Language Creation, Persona, and Template were chosen as they applied most to the calendar assistant use case and commonly found success.

The second paper used to find prevalent prompting techniques is titled ”A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications” published by Stanford University in early 2024 (2). Unlike the previous paper, Chadha et. al. only catalog various techniques and the research papers associated with them. While not organizing them by ”prompt patterns” the different techniques are still associated with which problems they attempt to solve. From this paper, few-shot, chain-of-thought, and self-consistency were chosen for their popularity and success's in similar tasks.

Finally two papers on chain-of-thought (3) and self-consistency (4) were referenced for further details on how to get the most out of each technique.

2 Methodology

2.1 Approach

The goal of the calendar assistant is an LLM that is capable of making modifications to a users calendar. For this reason we need a way to convert a list of calendar events into text. In the original project this was done through converting the calendar to ICAL format (Figure 3). Throughout that project we found that the ICAL format was incredibly complicated, with hundreds of attributes each with many ways to represent them. This made it more difficult for the model to consistently output valid calendars as well as accurately parse inputted parameters. Additionally, due to the amount of rules for the ICAL syntax, it would have been infeasible to prompt the model with the rules so we had to rely on the model's prior knowledge. For this reason we switched to a simplified JSON representation where each JSON would have exactly eight precisely defined attributes (Figure 4).

With this text based calendar representation, the plan was to prompt the model to be prepared for a calendar in this format in the input surrounded with <CAL></CAL> tags to separate it from the rest of the input. Additionally along with each response, the model would be prompted to include a modified version of the inputted calendar again in <CAL></CAL> tags. This decision was supported by through the Meta Language Creation and Template patterns (1).

2.2 Interface

The original deep learning project used a plain HTML website with text boxes for the input of the calendar as well as the request with the output of the model being printed to the screen (Figure 1). This format made it difficult to validate outputs of the model by hand as any calendar output needed to either be transferred to an application that could parse it or parsed by hand. For this reason a new interface was created using React. This new interface would not only include an improved chat box that could keep track of chat history but also a build in calendar UI allowing for the user to edit calendars and input them all in one place. Additionally calendar outputs from the model would be immediately reflected visually, making manual validation much easier.

2.3 Prompting Techniques

For this project I decided to compare five different prompting techniques: self-consistency, chain-of-thought, few-shot, template, and persona. Additionally, I included a naive prompt which included the meta language creation technique for describing the calendar to text format as well as a basic description of the task. Each other prompt would be the naive prompt along with modifications according to the research papers on how to implement the different techniques. Finally I also included a combination of the few-shot and the template technique as this was most similar to the prompt that was found most helpful in the original deep learning project.

2.4 Models

In the original deep learning project the only model used was GPT 3.5. To get more comprehensive results, in this project four different state of the art models were chosen: GPT-4 Turbo, Mistral Large, GPT 3.5, and Mistral 7b. GPT-4 Turbo and Mistral Large were chosen as they are both top performing large models. GPT 3.5 was added for comparison to GPT-4 and Mistral 7b was chosen to see how a smaller model performs in comparison.

2.5 Analysis

To evaluate the different prompts, I needed to generate a set of questions with corresponding valid answers. To this end I started by writing a few by hand, including input and output calendars. Then I had GPT-4 generate more in the same style which I then validated thought the prototype. Once the set of questions was, a validation pipeline was written in Python. This pipeline would be given a prompt along with a set of validation questions which would be fed into the model under test. The result of these questions would be organized and fed into GPT-4 along with ground truth answers to get a set of ratings. GPT-4 was prompted to provide scores in two categories, clarity and accuracy. Clarity corresponded to the natural language reasoning and weather or not it made sense in regards to the prompt. Accuracy corresponded to how similar the outputted calendar was to the ground truth calendar. Additionally, if the outputted calendar was invalid, this response would not be counted in the score and instead be tracked by ”hit rate.” This automated evaluation pipeline made it feasible to gather considerable amounts of data in a short amount of time.

3 Results & Discussion

When analyzing the results of all the different techniques, each prompt was compared to the naive prompt as a baseline. Figure 6 shows the average increase in performance from the naive baseline (across all models across clarity and accuracy). Unsurprisingly the prompt combining few-shot with template performed the best with an average one point advantage over the naive prompt. The template technique fits well with the structure necessary for a calendar assistant and the few-shot technique has shown the most success in the literature. Additionally, the best performing stand alone technique was self-consistency, an improvement on chain-of-thought for tasks that don't have definite answers. Another interesting detail revealed by this data is the persona technique performed lower than the naive baseline. Because there is no precedent for this happening in the literature I reviewed the only conclusion I can make is this was due to variation or improper implementation. Figure 7 shows the average increase in performance from the naive baseline in the best performing model for each prompt (across clarity and accuracy).This data would likely be the most useful if one wanted to further pursue a maximally accurate AI calendar assistant. This data matched the previous graph with few-shot plus template performing the best. Again persona performs worse than baseline here. Finally 8 compares the average score of each model. Unsurprisingly GPT-4, widely considered the most capable LLM, performs best. Interestingly Mistral Large slightly outperformed GPT 3.5 showing the capabilities of competing models. Then Mistral 7b unsurprisingly performs the worst as the smallest model by a large margin. Although with closer examination of its outputs, for such a small model its performance on a complex task was surprisingly promising.

4 Conclusions

In conclusion, this project successfully enhanced a calendar assistant by leveraging advanced prompt engineering techniques and implementing a more user-friendly interface. The analysis demonstrated that combining few-shot and template prompting techniques yielded the most significant improvement in the assistant's performance. Self-consistency also proved to be a valuable standalone technique. Interestingly, the persona technique did not perform as expected, indicating a need for further investigation.

The evaluation highlighted GPT-4's superior performance, with Mistral Large also showing promise, outperforming GPT-3.5. Despite its smaller size, Mistral 7b's performance was noteworthy for its capability on complex tasks. These findings underscore the potential of diverse models and sophisticated prompting strategies in enhancing AI-driven calendar assistants.

Given more time, it would have been beneficial to expand the scope of the evaluation by including a wider variety of questions and testing additional models, such as LLaMA 3. Additionally, running the validation multiple times for each set could provide a more robust and comprehensive understanding of the techniques' effectiveness. Future work could build on these insights to further refine and optimize AI calendar assistants for practical use.

5 References

[1] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," 2023.

[2] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," 2024.

[3] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models."

[4] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

A Appendix

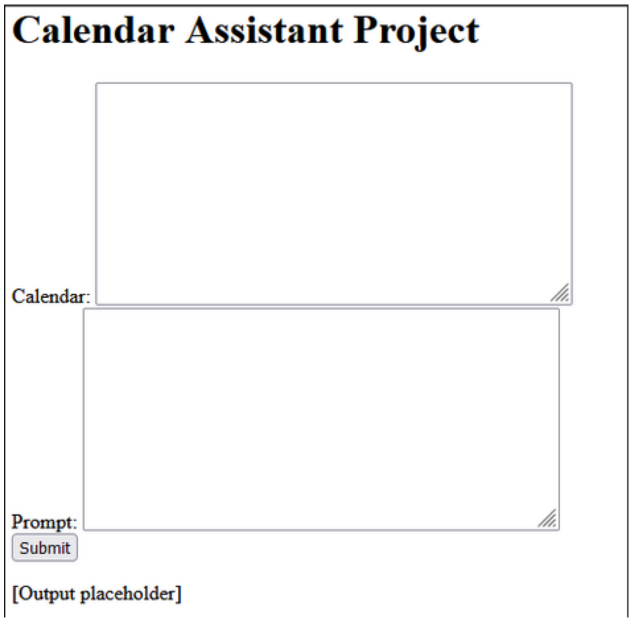


Figure 1: Old prototype

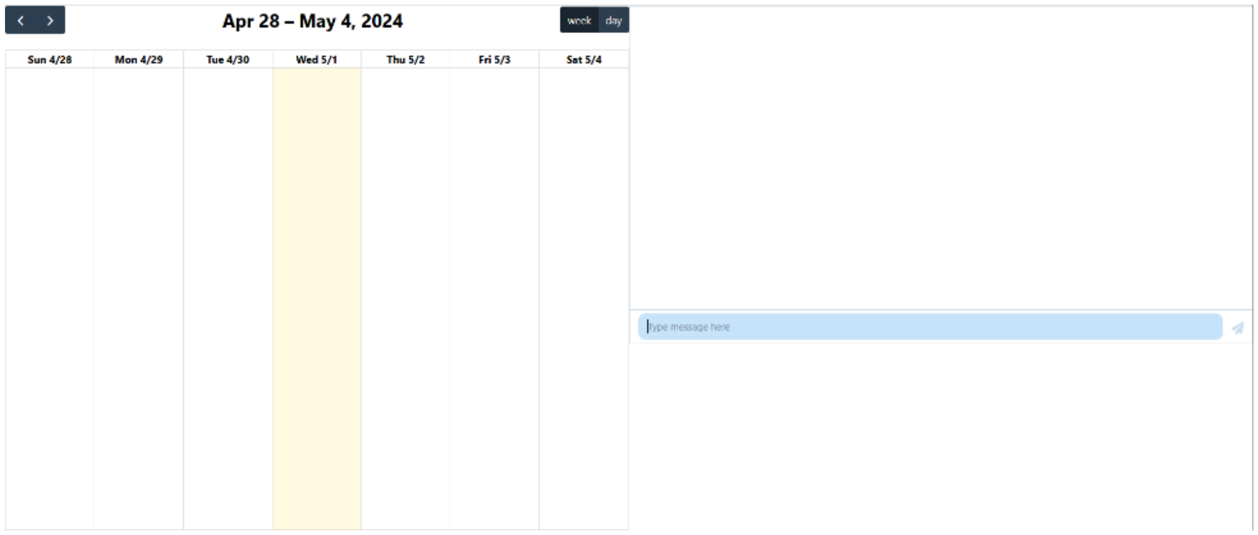
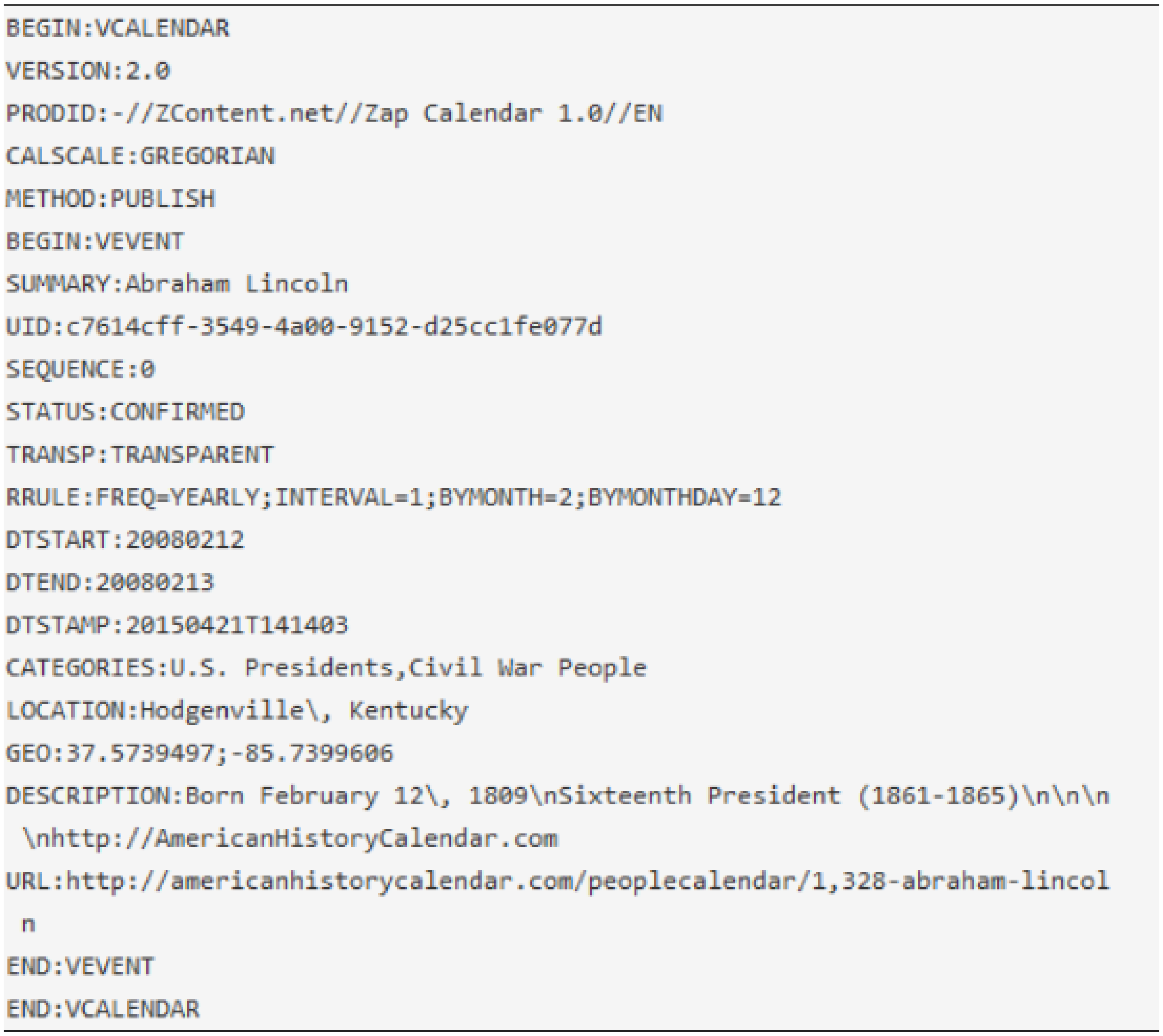
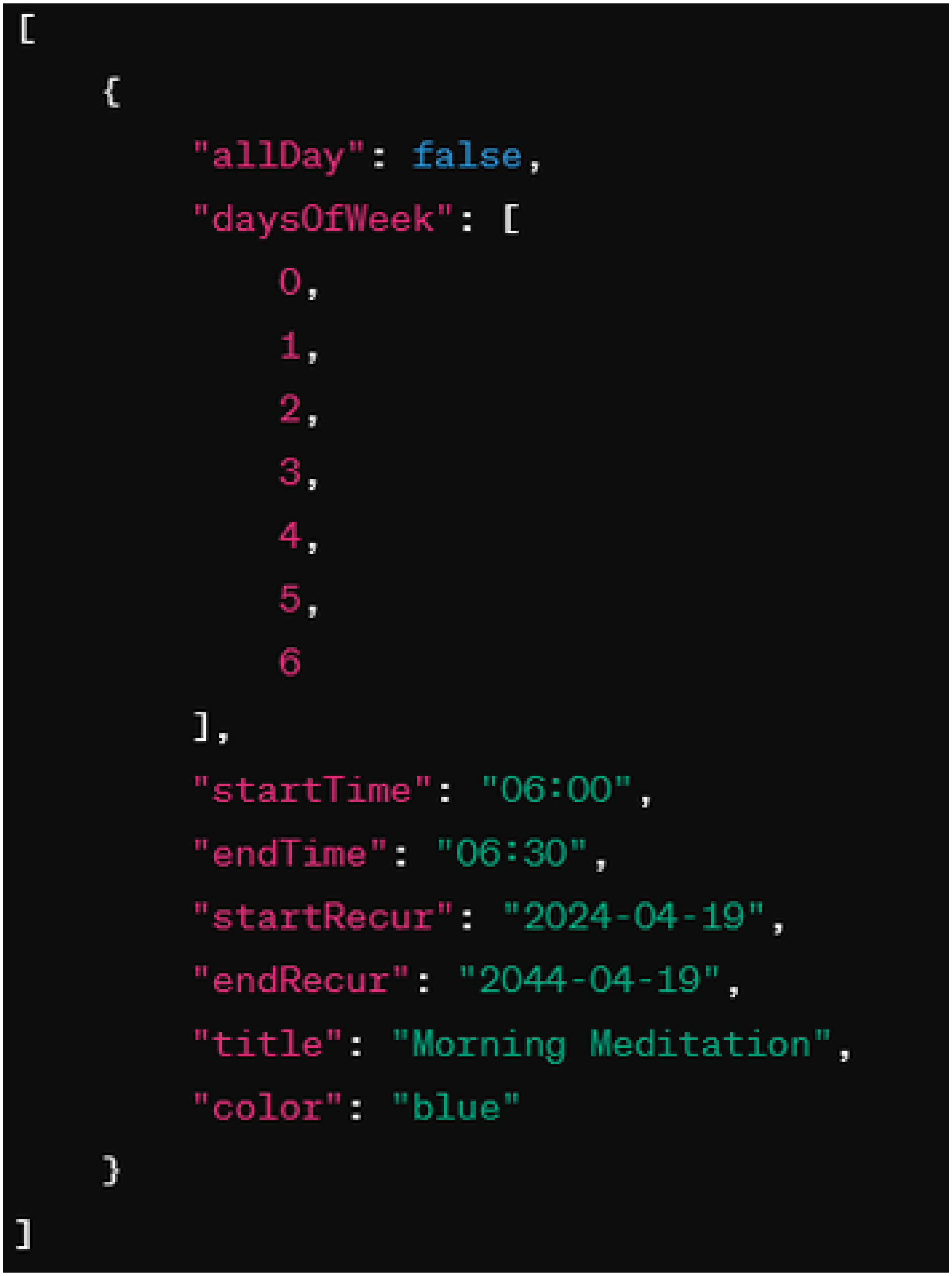


Figure 2: New prototype



```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//ZContent.net//Zap Calendar 1.0//EN
CALSCALE:GREGORIAN
METHOD:PUBLISH
BEGIN:VEVENT
SUMMARY:Abraham Lincoln
UID:c7614cff-3549-4a00-9152-d25cc1fe077d
SEQUENCE:0
STATUS:CONFIRMED
TRANSP:TRANSPARENT
RRULE:FREQ=YEARLY;INTERVAL=1;BYMONTH=2;BYMONTHDAY=12
DTSTART:20080212
DTEND:20080213
DTSTAMP:20150421T141403
CATEGORIES:U.S. Presidents,Civil War People
LOCATION:Hodgenville\, Kentucky
GEO:37.5739497;-85.7399606
DESCRIPTION:Born February 12\, 1809\nSixteenth President (1861-1865)\n\n\n
 \nhttp://AmericanHistoryCalendar.com
URL:http://americanhistorycalendar.com/peoplecalendar/1,328-abraham-lincol
 n
END:VEVENT
END:VCALENDAR
```

Figure 3: Example of an event in ICAL format



```
[
    {
        "allDay": false,
        "daysOfWeek": [
            0,
            1,
            2,
            3,
            4,
            5,
            6
        ],
        "startTime": "06:00",
        "endTime": "06:30",
        "startRecur": "2024-04-19",
        "endRecur": "2044-04-19",
        "title": "Morning Meditation",
        "color": "blue"
    }
]
```

Figure 4: Improved text calendar representation

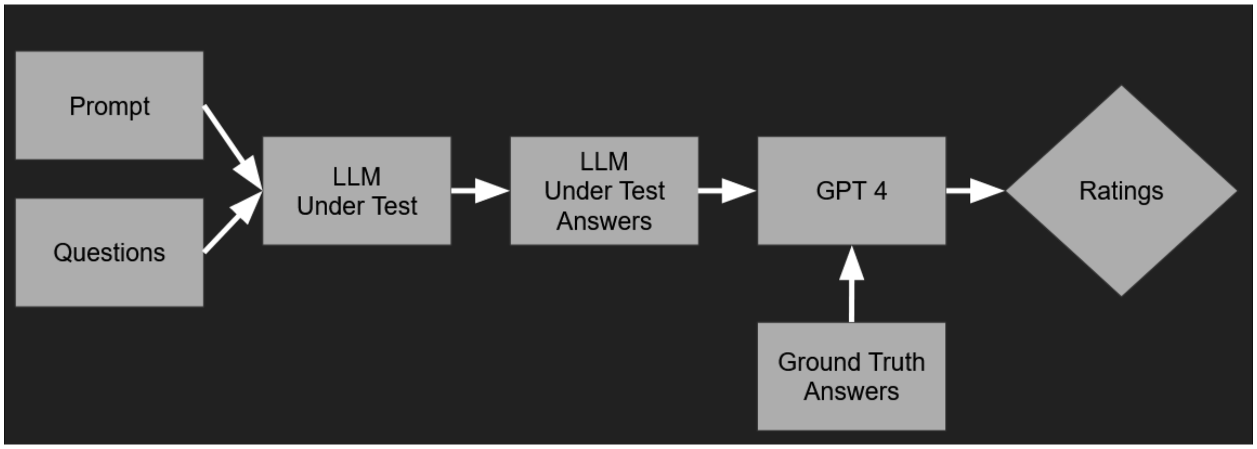


Figure 5: Diagram of the evaluation pipeline

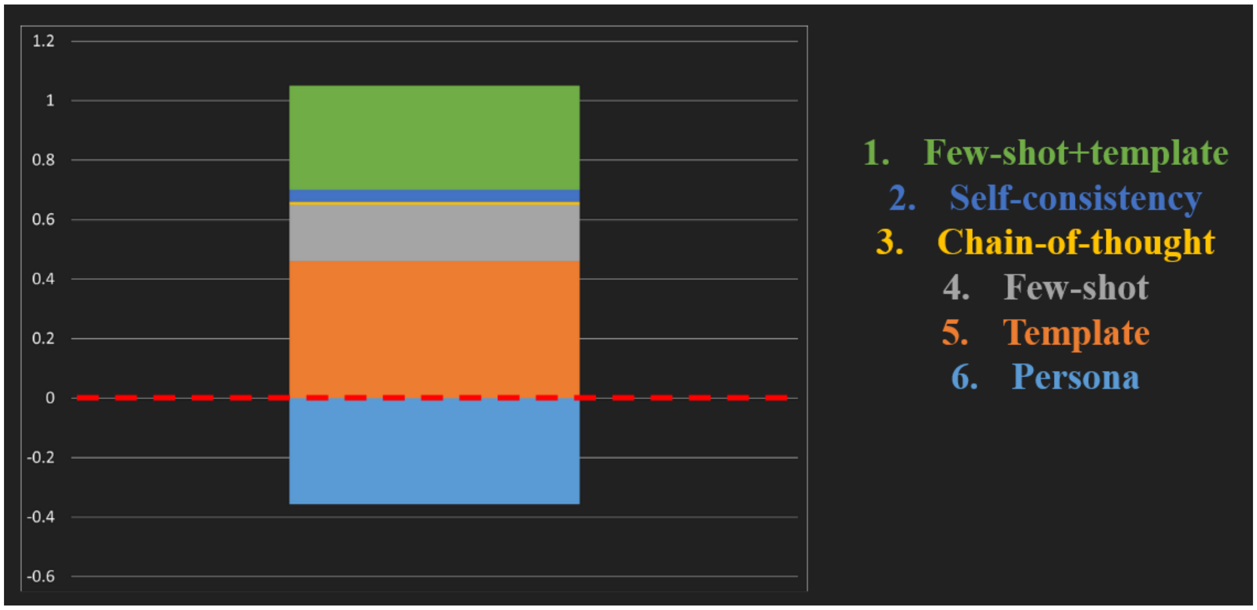


Figure 6: Improvement in average score

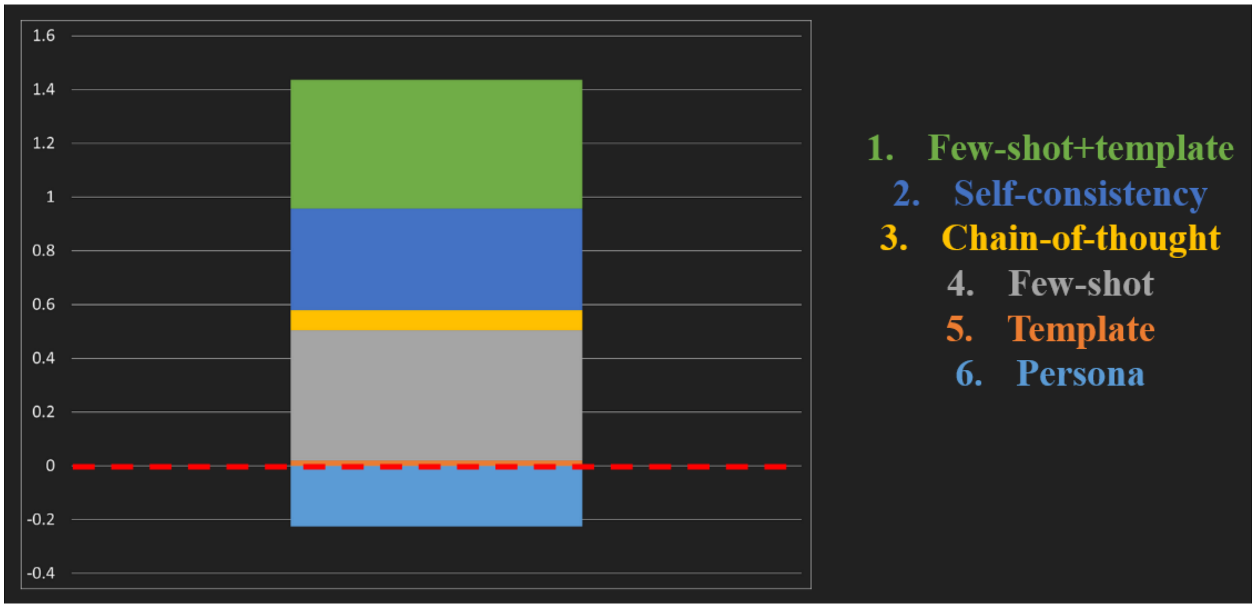


Figure 7: Improvement in average score in the best performing model

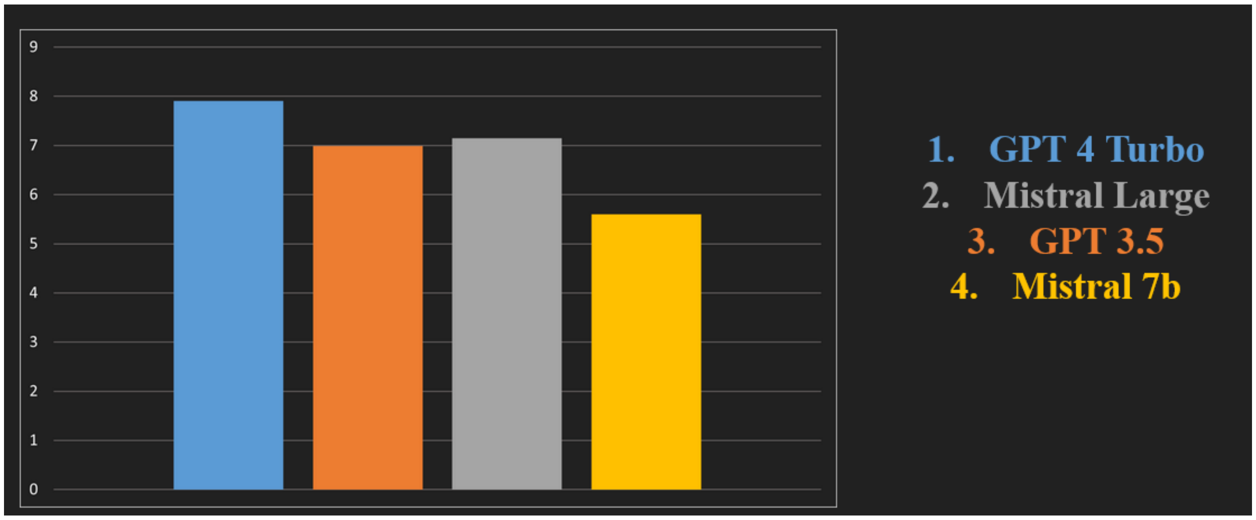


Figure 8: Average scores of each model