Developing Stratagies for the Bidding Card Game "Diamonds" with GenAI

Radhika Amar Desai

March 26, 2024

1 Introduction

This report aims to teach the GenAI tool "Gemini" how to play a previously unknown game, namely the "diamonds" bidding card game, and develop a strategy for bidding on the cards. We first teach the game to GenAI and develop strategies for the same. Gemini has been used as a coding assistant to code the game as well as the strategy developed. The codes given by GenAI were corrected through repetitive prompting and interference of the programmer. For developing the bidding strategy, the tool was used to refine the programmer's idea.

The game features a dynamic interplay of four distinct card suits, with the banker exclusively holding cards from the diamond suit, while other players receive cards from the remaining three suits. Each round unfolds through a meticulous bidding process, where players sequentially offer cards while the banker reveals a diamond suit card. The pivotal moment arises during bid evaluation, where card values, irrespective of their suits, determine the victor. The highest bid secures points equivalent to the displayed diamond card's numerical or character value. In cases of tied bids, points associated with the revealed diamond card are equitably distributed.

The game adheres strictly to a hierarchical value system, where numeric cards maintain their face value, followed by Jack, Queen, King, and Ace in ascending order.

2 Problem Statement

The objective of this project is to develop the "diamonds" game code, create and code a bidding strategy using GenAI Tools for assistance. The project aims to analyze how GenAI Tool responds to various prompts, whether it can retain the context of the discussion, how well it learns, and the quality of code generated by the GenAI tool "Gemini". The learning objectives of this project are understanding LLMs and mastering prompt engineering.

3 Teaching GenAI the Game

This section describes how the "Diamonds" game was taught to GenAI and how GenAI was used to write Python code for the game. The section analyzes Gemini's responses during the process of learning and coding the game.

The rules of the game were explained to Gemini. Gemini examined the rules and summarized its understanding of the game. Initially, Gemini associated the value of a card with the suite it belongs to, despite the opposite being clearly stated in the rules. This misunderstanding was easily clarified through further prompting, and Gemini quickly understood its mistake. The understanding of the game was further tested by asking Gemini to give demonstrations of the game. Any misunderstandings found were corrected through further prompting, and Gemini corrected its mistakes quickly.

Gemini generated a multi-file structure for the game once, even when it was not asked to do so. However, when asked to code in a single Jupyter notebook, Gemini generated code organized by appropriate classes, objects, and methods. Mistakes in the code were corrected through further prompting and user interference.

Some notable aspects of Gemini's behavior were:

- It quickly corrected its mistakes.
- It generated a thorough code structure without being asked.
- It did not ask for any clarifications before generating code.

4 Developing the Bidding Strategy

This section outlines the process of developing and coding a bidding strategy in Python using GenAI, along with an analysis of Gemini's responses during this task.

Gemini was tasked with proposing strategies for the game, presenting three types: basic, intermediate, and advanced. The programmer selected one of the advanced techniques, the probabilistic bidding strategy, and asked Gemini for a detailed breakdown. Gemini described its approach, which involved calculating expected points by multiplying the probability of a diamond card appearing with the value of that diamond card. However, a flaw was noticed in Gemini's formula as it only considered a victory if the player's card value exceeded the revealed diamond card's value, which was corrected through further instructions.

Gemini then generated an excellent code structure with appropriate functional decomposition for the strategy. However, the code initially produced was incorrect, requiring further prompting and user intervention for corrections. Despite initially struggling due to its misunderstanding of victory conditions, Gemini eventually corrected its mistake but continued to exhibit stubbornness in subsequent prompts, particularly when asked to use dictionary composition.

The programmer had to intervene significantly in developing the code for this part, with Gemini providing suggestions and corrections but only stopping

when explicitly asked to do so. Next, Gemini was asked to upgrade its strategy by considering the cards of other players, which again required substantial user intervention to align with the correct perception of victory conditions.

At this stage, Gemini was utilized primarily as a coding tool, being asked to code specific functions with well-defined function definitions. It performed well in valuing each card based on its probability of victory and even corrected the formula for calculating this probability.

Notable observations about Gemini's behavior include:

- Correcting mistakes only in response to the next prompt.
- Providing unique suggestions and logical corrections.
- Offering suggestions and warnings even when corrections were not initially appreciated.

5 Conclusion: Developing an Effective Prompting Strategy

This section presents an analysis of Gemini's behavior based on the conducted experiment and suggests an effective prompting strategy for similar projects.

Gemini consistently provides a professional code structure, making it a valuable tool for development. From the observations made, we can divide our prompting process into two phases: the explanation phase and the coding phase.

During the explanation phase, corrections and clarifications provided to Gemini tend to be retained for a longer duration. For instance, when teaching the card game to Gemini, corrections made during this phase were remembered well. However, during the coding phase, corrections related to the problem statement itself were not retained for an extended period. This was evident when developing the bidding strategy.

Therefore, for GenAI tools like Gemini, it is crucial to invest sufficient time in explaining the problem statement thoroughly and verifying their understanding with multiple examples. During the coding phase, it is beneficial to proceed step by step. Initially, focus on developing a code structure with Gemini, and then proceed to code individual components such as classes, objects, or functions.

Additionally, it is advisable to revisit and explain the problem statement multiple times, especially when transitioning to a different part of the problem (e.g., developing a bidding strategy). This is because Gemini tends to forget previous conversations, and repetitive explanations can help reinforce understanding and reduce errors.

In summary, an effective prompting strategy for GenAI tools like Gemini includes thorough explanation and verification during the explanation phase, step-by-step coding with a focus on structure first, and regular reinforcement of the problem statement understanding throughout the coding process.

6 References

The attached link contains a GitHub repository consisting of transcripts of conversations with Gemini and Python code for the card game.

- GitHub: genAI.git