Prompt Engineering ChatGPT for Codenames

Matthew Sidji
Faculty of Engineering and Information Technology
The University of Melbourne
Melbourne, Australia
msidji@student.unimelb.edu.au

Matthew Stephenson
College of Science and Engineering
Flinders University
Adelaide, Australia
matthew.stephenson@flinders.edu.au

***Abstract*—The word association game Codenames challenges the AI community with its requirements for multimodal language understanding, theory of mind, and epistemic reasoning. Previous attempts to develop AI agents for the game have focused on word embedding techniques, which while good with other models using the same technique, can sometimes suffer from brittle performance when paired with other models. Recently, Large Language Models (LLMs) have demonstrated enhanced capabilities, excelling in complex cognitive tasks, including symbolic and common sense reasoning. In this paper, we compare a range of recent prompt engineering techniques for GPT-based Codenames agents. While there was no significant game score improvement over the baseline agent, we did observe qualitative changes in agents' strategies suggesting that further refinement has potential for score improvement. We also propose a revised Codenames AI competition specifically focusing on the use of LLM agents.**

***Index Terms*—Codenames, ChatGPT, Prompt Engineering, Game Playing Agents, AI**

I. Introduction

Games have long been favoured as test beds for Artificial Intelligence (AI) due to them providing controlled and well defined environments suitable for developing and benchmarking AI capabilities. Previously, perfect information competitive games such as Chess and Go have been conquered by AI's such as Deep Blue and AlphaGo [1], [2]. Recently, games requiring cooperation, natural language understanding, or theory of mind have proved challenging for AI to master [3], [4]. One game in particular that captures all of these challenges is the word association game Codenames [5].

Codenames is a popular language-based cooperative game in which teams of players have to discern relationships between words using single word clues given by their teammates. Two teams - each with a codemaster and guesser - share a board comprising of 25 words. The codemaster is secretly given their teams target words, and must deliver a series of one-word clues that help the guesser correctly identify these. Previous approaches to develop Codenames AI have shown good performance when paired agents are using the same word association strategy [6]. While there is some work on developing more adaptable models [7] the majority of the models exhibit lowered performance when playing with teammates using a different approach.

Large Language Models (LLMs) known for their flexibility in natural language understanding across many domains, offer a potential solution to this limitation. The recent explosion in popularity of LLMs has seen them being applied to many domains [8], [9], yet there is little work using them for Codenames. This is surprising given the many emergent capabilities displayed by LLMs being useful for performance in Codenames. In particular, evidence of theory of mind reasoning emerging from LLMs shows potential for their use in Codenames, which requires an understanding of how a teammate might interpret or formulate a clue [10]. The associated burgeoning field of prompt engineering is also seeing a wealth of techniques to improve LLM performance in different tasks. Investigating the impact of various prompting techniques on performance is more accessible to novice programmers, as it primarily involves using natural language without the need to train new models. This provides an ideal opportunity to develop diverse AI agents from both academic researchers and novice enthusiasts alike, potentially sparking widespread interest and competition.

II. Related Work

In this section we will discuss the previous approaches used to developing Codenames AI agents, as well as recent developments in the field of prompt engineering that might provide promising avenues to utilise in Codenames agents.

A. Codenames AI

Codenames has garnered interest from the AI community due to it requiring multi-modal language understanding, asymmetric cooperation, theory of mind, and epistemic reasoning [6]. Kim et al. (2019) introduced the first Codenames competition with agents using word embedding techniques. They found models paired with themselves achieved 100% accuracy, but achieved much lower results when paired with other teammates. Jaramillo et al. (2020) extended upon this work, developing agents which utilised term frequency - inverse document frequency (TF-IDF), Naive-Bayes, and the GPT-2 Transformer model. They found that the transformer model, when compared to bots developed by Kim et al. (2019), met or exceeded performance in all metrics, with the transformer model being preferred as codemaster (clue giver) when tested with human participants [11].

Further tests with human evaluators were done by Koyyalagunta et al. (2021), who developed multiple methods to improve Codenames AI performance for knowledge graphs, word embeddings and knowledge-based methods. As they

were testing with human evaluators, their aim was to create agents that present more human interpretable clues, as word embedding agents would sometimes give nonsensical but correctly guessed clues when paired with another word embedding guesser [12]. Other researchers have focused on agents that can adapt to their teammate. Archibald et al. (2024) created the Adaptive Codenames Ensemble (ACE) which changes which base Codenames agent it gives clues with depending on how their teammate guesses [7]. Despite the high focus on word embedding techniques, recent work shows the potential of prompting techniques for improved Codenames performance. Ozturkler et al. (2023) applied the LLM prompting technique ThinkSum to promote deductive reasoning in Codenames clue guessing. They achieved a score roughly 20% better than few-shot prompting for Codenames guesser agents [13]. This promising start in exploring prompt engineering techniques for Codenames, given the increased focus on prompt engineering and LLMs, indicates significant potential for further research in developing Codenames agents.

B. Prompting Techniques

Prompt engineering, a burgeoning field within AI research, focuses on enhancing LLM response quality by refining input prompts. The field has recently seen significant growth, producing many innovative techniques grounded in cognitive theories that emulate human problem-solving strategies, such as subdividing problems or engaging in self-reflection [14], [15].

These techniques have been shown to be effective in boosting LLM performance across tasks including common sense, arithmetic, and symbolic reasoning, and serve as valuable benchmarks for LLM capabilities [16]. The most popular among these, Chain-of-Thought (CoT) reasoning, introduced by Wei et al. (2022), provides models with few-shot exemplars and prompts them to split tasks into subproblems and step through them sequentially. This was shown to significantly enhance task performance even in zero-shot scenarios by simply appending "Show me step by step" to the prompt [17], [18]. Building on CoT, Wang et al. (2023) develop a self-consistency approach that selects the most consistent response from multiple reasoning paths, while Yao et al. (2023) proposed Tree-of-Thought (ToT) which employs heuristic searches across multiple paths [14], [19]. These and similar techniques improve model performance for tasks that require non-linear thinking or multiple variables [20].

Additional techniques that mimic the way humans iteratively refine pieces of writing or learn through self-explanation have proven effective. These ask models to follow a similar process of self critique or explanation and further improve performance for certain tasks compared to CoT prompting [15], [21]. Additionally, Wang et al. (2024) introduced Solo Performance Prompting (SPP), which employs a committee of personas to evaluate various solutions, enhancing the robustness and accuracy of responses [22].

While all of these techniques have shown superior performance in some tasks solving more complex tasks often involves using a combination of these techniques [20]. As identified by Srivastava et al. (2022), generating a clue in Codenames is considered a "composite" task which requires several distinct skills [16]. It is therefore unclear which technique - or combination of techniques - would be best suited for the problem of giving and guessing Codenames clues.

III. Methodology

Using the pre-existing Codenames competition framework, we developed codemaster (clue giver) and guesser agents that utilise OpenAI's GPT-4 model. Both agents are provided with the same base description of the game's rules as an initial system prompt, along with whether they are playing as the codemaster or guesser. For the codemaster agent, we implemented six versions with different prompt-engineered instructions. These agent versions are summarised below, with full prompts and game rules available in our public code repository.[1]

1) Default: This agent operates as a baseline for comparing other prompt-engineering approaches. The codemaster is provided with the remaining words and their associated labels (red, blue, civilian or assassin); and is asked to provide a single clue (word and required number for the guesses) in a precise format.

2) Cautious: This agent is an extended version of the Default agent with an additional prompt instructing it to always provide the number one for the number for the guesses in its clue. The intention here is to create a cautious agent that only needs to find an association with a single red word.

3) Risky: This agent is an extended version of the Default agent with an additional prompt instructing it to pick a large number for the number for the guesses in its clue. The intention here is to produce a risky agent that tries to find associations between many red words.

4) Chain of Thought: This agent is an extended version of the Default agent that additionally utilises the zero-shot CoT technique discussed in [18]. This is done by appending the default prompt with "Solve the task step by step". This results in the agent responding in a stepwise fashion, explaining associations between a subset of target words, suitable candidate clues for those words, and how these are unrelated to non-target words.

5) Self Refine: This agent uses a series of three responses to give a refined final answer, utilising the technique outlined in [21]. This involves instructing the agent to give an initial clue, in the same manner as the Default agent. The agent is then instructed to give feedback on this initial clue, critiquing it on the basis of its relation to the red words and low likelihood of associations with blue, assassin and civilian words. Finally the agent is then instructed to give a second clue but incorporating the previous feedback. Ideally this agent is able to pick up on any errors in makes in the initial clue, improving upon them in the second attempt.

[1] https://github.com/stepmat/Codenames_GPT/tree/CoG_2024

TABLE I
SUMMARY STATISTICS FOR ALL AGENT VERSIONS ACROSS 50 GAMES

Agent	Mean	Median	Min	Std Dev	Loss
Default	9.86	7	5	6.78	16%
Cautious	9.80	9	8	3.99	6%
Risky	12.00	8	4	8.29	28%
Chain of Thought	10.92	7	4	8.08	24%
Self Refine	9.28	7	5	6.49	14%
Solo Performance	11.18	7	4	7.92	24%

6) Solo Performance: This agent uses the Solo-Performance Prompting (SPP) technique outlined in [22]. After receiving the game's rules and current board state, the agent is prompted to internally construct multiple personas that help to formulate an appropriate clue. The agent then produces a back and forth dialogue between these personas, with each proposing or criticising clues until a consensus is reached. This method also includes a small number of few-shot examples, in order to have the agent perform SPP in the correct format.

IV. EXPERIMENTS

To evaluate the performance of GPT-4 for playing Codenames, as well as the effectiveness of each prompt engineering technique, we played 50 individual games for each codemaster version using the available Codenames competition framework. Both the codemaster and guesser agents utilised the 'gpt-4-1106-preview' model from OpenAI, with all model hyperparamters remaining at their default values. Full results for these experiments are available in the provided code repository, with summary results and qualitative observations described below.

A. Results

Table I provides a summary of how each agent performed across the 50 experiment games. Mean, Median, Min and Standard Deviation (Std Dev) each relate to the final score of the game. In this case, a lower score indicates a better performance. The Loss column refers to the percentage of games that ended in a loss. For the current Codenames competition rules a loss gives a score of 25 points, meaning that losing a game often has a disproportionately high impact on the Mean and Std Dev results.

V. DISCUSSION

Based on the results of our experiments, none of the implemented prompt engineering techniques produced a significant performance improvement over the Default agent. However, we were able to observe several differences in how these agents played.

Looking first at the Cautious and Risky agents, these demonstrated highly divergent playstyles. The Cautious agent would always choose the number one for its clues, leading to a minimum score of 8 (the exact number of words that needed to be selected) as well as a low chance of losing. The Risky agent gave the opposite behaviour, having a low minimum score but also an increased number of losses. Given that a loss has such a high impact on the final score, it would appear that a cautious approach may be a more beneficial strategy for the current version of the game.

The Chain of Thought and Solo Performance agents appeared to favour higher clue numbers, perhaps overthinking the problem and leading to a riskier style of play. In comparison, the Chain of Thought agent tended to revise the number provided alongside its clue down if the provided feedback indicated skepticism or hesitancy about the initial clue. This likely worked in the Self Refine agent's favour, as our earlier results already demonstrated that a more cautious style of play tends to lead to a better overall score.

A. Qualitative Observations

Observing the reasoning given by different versions of our codemaster agents reveals the provided clues are often closely associated with one of the target words, but that the agent will claim a connection to other target words that is highly tenuous. For example, when the Solo Performance agent gave the clue ('OCEAN', 2) it claimed that "This clue should guide the guesser to BEACH and TRUNK (as in the trunk of a car often bringing items to the beach)". Agents would also often inaccurately consider the potential relationship of their clue to other non-target words, which is especially important for the assassin word. For example, when the Chain of Thought agent gave the clue 'DINNER' it reasoned that "DINNER does not give a clue towards the assassin word KNIFE that is actually related to the dinner setting, but not typically described by DINNER itself".

B. Prior Codenames Agents

When comparing the performance of our GPT agents to prior agent results presented in [6], [11], it would appear that we are not able to match their performance. However, all of these prior agents have been developed for the exact set of 395 available words currently present in the game, and would be unable to operate successfully if new words were added. Our presented agents do not have this restriction, and are able to play using a completely new set of words without any additional training. Many of these prior agents also rely on identical codemaster-guesser agent pairs to perform well. While our experiments do use the same GPT-4 base model for the codemaster and guesser agents, they are not designed with this matching in mind.

C. Competition Rule Deviations

The current Codenames competition framework deviates from the original Codenames rules in two important aspects. Firstly, the game has been reworked into a purely cooperative game, where only the red team gives clues and makes guesses. As a result, the blue and civilian tiles function almost identically, with the only difference being that the red team loses if they select all the blue tiles (which is highly unlikely). Secondly, the guesser cannot deviate from

the number of guesses specified in the codemaster's clue. In the original game's rules, the guesser can specify any number of tiles between one and one more than the number of guesses specified by the codemaster (e.g. if the codemaster's clue specifies the number 3, the guesser can select 1-4 words).

Both of these rule changes have a significant impact on how the game is played, removing elements that contribute to the game's strategic depth and social deduction aspects. Having a second team increases the negative impact of picking a tile with the opposing teams colour and provides additional information about the remaining tiles on the board (via the clues and guesses made by the opposing team). Permitting the guesser to deviate from the number of guesses also adds to the strategy of the game, allowing clue information to be reused over multiple turns (in the event of an incorrect guess) and for the guesser to stop early if they are unsure about the codemaster's clue.

VI. Future work

The main catalyst for this preliminary study into the effectiveness of GPT models for playing Codenames was the desire to organise a new Codenames AI competition based on the original game rules (competitive teams of agents rather than purely cooperative) with a renewed focus on LLM agents and student submission accessibility. The authors of this paper intend to develop a new Codenames framework based on the original game's rules over the coming months, and submit an application for an AI agent competition using this framework to the IEEE Conference on Games (CoG) conference in 2025. Agents for this competition will also be evaluated on a previously unseen set of tile words, increasing the emphasise on general AI reasoning. We feel that the recent resurgence in LLM advancements and interest over the past few years, the competition will have a wide appeal to members of the Game AI research community.

From the prompt engineering side, we intentionally kept our prompts as broad and general as possible for the engineering techniques used. However, its likely that additional prompts focused on addressing each agent's weaknesses (such as adding and "assassin skeptic" role for the Solo Performance agent) could help to reduce the number of losses and increase overall performance. Prompt engineering techniques could also be applied to the guesser agent as well as the codemaster. We hope that the aforementioned AI competition will help to increase research into developing and applying new prompt engineering approaches for Codenames.

VII. Conclusion

Within this paper, we investigated the effectiveness of GPT-4 for providing suitable clues and guesses for the word-based game Codenames. We also explored the impact that several basic prompt-engineering techniques had. While none of these significantly improved the performance of our default agent, they did appear to affect the agent's overall playstyle. We feel that word games such as Codenames offer a unique challenge and performance benchmark for advanced LLM models, and hope to organise a revised AI competition focused on developing Codenames agents.

References

[1] M. Campbell, A. J. Hoane, and F.-h. Hsu, "Deep Blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, 2002.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–, Oct. 2017.

[3] R. Canaan, H. Shen, R. Torrado, J. Togelius, A. Nealen, and S. Menzel, "Evolving Agents for the Hanabi 2018 CIG Competition," in *IEEE Conference on Computational Intelligence and Games (CIG)*, 2018.

[4] R. D. Gaina and M. Balla, "TAG: Pandemic Competition," in *2022 IEEE Conference on Games (CoG)*, 2022, pp. 552–559.

[5] V. Chvátil, *Codenames*. Czech Games Edition, 2015.

[6] A. Kim, M. Ruzmaykin, A. Truong, and A. Summerville, "Cooperation and Codenames: Understanding Natural Language Processing via Codenames," vol. 15, no. 1, pp. 160–166, 2019.

[7] C. Archibald and S. Brosnahan, "Adapting to teammates in a cooperative language game," *arXiv*, 2024.

[8] G. Todd, S. Earle, M. U. Nasir, M. C. Green, and J. Togelius, "Level generation through large language models," in *Proceedings of the 18th International Conference on the Foundations of Digital Games*. Association for Computing Machinery, 2023.

[9] Q. Wan, S. Hu, Y. Zhang, P. Wang, B. Wen, and Z. Lu, "It Felt Like Having a Second Mind: Investigating Human-AI Co-creativity in Prewriting with Large Language Models," *arXiv*, 2023.

[10] M. Kosinski, "Theory of mind may have spontaneously emerged in large language models," *arXiv*, 2023.

[11] C. Jaramillo, M. Charity, R. Canaan, and J. Togelius, "Word autobots: Using transformers for word association in the game codenames," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, pp. 231–237, Oct. 2020.

[12] D. Koyyalagunta, A. Sun, R. L. Draelos, and C. Rudin, "Playing Codenames with Language Graphs and Word Embeddings," *Journal of Artificial Intelligence Research*, vol. 71, pp. 319–346, 2021.

[13] B. Ozturkler, N. Malkin, Z. Wang, and N. Jojic, "ThinkSum: Probabilistic reasoning over sets using large language models," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2023, pp. 1216–1239.

[14] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 11 809–11 822, 2023.

[15] H. Gao, T.-E. Lin, H. Li, M. Yang, Y. Wu, W. Ma, and Y. Li, "Self-Explanation Prompting Improves Dialogue Understanding in Large Language Models," *arXiv*, 2023.

[16] S. et al., "Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models," *arXiv*, 2023.

[17] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[18] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 199–22 213, 2022-12-06.

[19] 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*, 2023.

[20] O. Fagbohun, R. M. Harrison, and A. Dereventsov, "An Empirical Categorization of Prompting Techniques for Large Language Models: A Practitioner's Guide," *arXiv*, 2024.

[21] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-Refine: Iterative Refinement with Self-Feedback," *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 534–46 594, 2023.

[22] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji, "Unleashing the Emergent Cognitive Synergy in Large Language Models: A Task-Solving Agent through Multi-Persona Self-Collaboration," *arXiv*, 2024.