Solving Wordle with Large Language Models

Jinwoo Baik
jbaik23@g.ucla.edu
Dang Nguyen
dangnth@g.ucla.edu
Sara Khosravi
sarakhosravi@g.ucla.edu
Lucas Bandarkar
lucasbandarkar@cs.ucla.edu

While Large Language Models (LLMs), such as ChatGPT, have demonstrated significant success across various language understanding tasks, recent evaluations have revealed notable shortcomings in their performance on specific tasks like solving Wordle puzzles. This study investigates the performance of various open-source LLMs on Wordle puzzles. To achieve this, we developed a chat system to benchmark LLM performance on Wordle, incorporating sophisticated prompts that include game explanations, few-shot demonstrations, and valuable feedback. We also designed automatic evaluation metrics to assess the ability of models to learn and adapt from previous guesses. Our findings indicate that open-source LLMs perform poorly on Wordle and often fail to learn from mistakes, underperforming even random guessing in some metrics. This project highlights the challenges and potential solutions for improving LLM performance on specific tasks, contributing to a deeper understanding of how these models can be optimized for downstream applications[1].

1 Introduction

Large Language Models (LLMs) have achieved remarkable success in a variety of tasks, ranging from machine translation to conversational AI (Brown et al., 2020). These models are capable of understanding and generating human-like text, making them valuable tools in fields like content creation, customer service, data analysis, and education (Naveed et al., 2023). Their ability to process vast amounts of text data from diverse sources enables them to perform complex tasks with high levels of accuracy and efficiency.

However, the recent performance of ChatGPT in solving Wordle puzzles has highlighted significant limitations in the capabilities of LLMs when applied to specific tasks. Despite its impressive prowess in a wide range of natural language processing tasks, ChatGPT-4 struggled considerably with Wordle[2], as reported by (Madden, 2024). The study underscored ChatGPT's surprising difficulty in accurately guessing words, even when partial information about correct letters was provided. This performance gap raises important questions about the real-world applicability of LLMs for tasks requiring nuanced understanding and precise language processing, such as word games.

Given that ChatGPT-4 is trained on extensive text data, including sources like Wikipedia and scientific articles, its underperformance in Wordle is particularly intriguing. Tests revealed inconsistencies in its ability to match patterns and frequently proposed incorrect words, indicating fundamental limitations in how LLMs represent and manipulate words in a game-like context. This suggests that while LLMs excel in many language-related tasks, they may still struggle with specific applications that require a different approach to understanding and processing words.

In this project, our objective is first, to investigate the capabilities of various open-source LLMs in solving Wordle puzzles; and second, to introduce new automatic metrics for evaluating the performance of these models. By exploring the performance of different LLMs on Wordle, we aim to identify potential improvements and modifications that can enhance their effectiveness. In addition, while existing metrics, such as the number of correct letters guessed and the number of attempts required to guess the word correctly, provide a starting point, there is a pressing need for more sophisticated metrics that can offer a deeper understanding of LLM performance on Wordle puzzles.

Our contributions in this project are threefold:

1. We build a chat system to benchmark the performance of LLMs on Wordle puzzles. To

[1]Code and data is available at https://github.com/jbaik1/CS-263-Wordle.git.

[2]https://www.nytimes.com/games/wordle/index.html

further improve the performance, we design an informative and sophisticated prompt consisting of game explanation, few-shot demonstrations, and valuable feedbacks.

2. We develop automatic evaluation metrics to measure LLM performance on Wordle. These include assessments of the quality of guesses, the consistency of pattern matching, and the ability to learn and adapt from previous guesses.

3. Our experiments reveal that, similar to ChatGPT, open-source LLMs also perform poorly on Wordle. In addition, some models fail to learn from mistakes, thus, underperform random guessing.

In summary, this project aims to shed light on the challenges and potential solutions for improving LLM performance in solving Wordle puzzles. By investigating various open-source LLMs and introducing new evaluation metrics, we hope to contribute to the broader understanding of how these models can be refined and optimized for specific downstream tasks.

2 Related Works

ChatGPT. Brown et al. introduced GPT-3, has proven to be a powerful language model which had strong performance in a variety of NLP tasks. This involves the contexts of zero-shot, one-shot, and few-shot learning. Its public successor ChatGPT, which is based on GPT 3.5 (Liu et al., 2023), have also showed exceptional performance in NLP tasks. However there are still limitations.

Solving Wordle using LLMs. Madden showed that ChatGPT struggled to solve Wordle puzzles. Zaidi built an interface for GPT 3.5 and GPT 4 to play Wordle game. Their conclusion was that GPT 4 are capable of solving Wordle while GPT 3.5 fails to do so. Both these two works did not present any quantitative results. Thompson compared the performance of GPT 4 with random solver. They showed that GPT 4 outperforms random solver by a clear margin. However, the number of games they tested was limited (max 100). In addition, while the accuracy of correct word is high for GPT 4, this is actually a result of allowing more guesses (10 instead of 5 in the standard setting). Furthermore, the evaluation metrics are limited to the number of correct guessed words and the number of guesses. In contrast to our work, all of the above works did not consider any other LLMs beside GPT family and the number of evaluation metrics is limited.

LLMs for solving other puzzles. There are many recent works investigating the capabilities of LLMs to puzzle solving. Deterministic rule-based puzzles, such as Sudoku, follow clear rules and have one specific solution is one of the most popular tasks for LLMs. Noever and Burdick fine-tuned GPT-2 (Radford et al., 2019) using 1 million Sudoku games to improve the model performance. Long introduced Tree-of-Thought (ToT) framework to solve complex reasoning tasks and showed a high success rate for solving Sudoku puzzles. Ishay et al. delved into neuro-symbolic approaches, demonstrating that well-prompted LLMs can generate answer set programming rules for Sudoku. In contrast to deterministic games, stochastic rule-based puzzles consists of random and hidden states making it more challenging for LLMs. Efforts on stochastic games include exploring LLMs for playing Minesweeper (Li et al., 2023), Poker (Gupta, 2023; Huang et al., 2024), and Werewolf (Xu et al., 2023). Different from both the aforementioned rule-based types, rule-less puzzles are the dominant category in the real world. Research on this category mainly focused on introducing new datasets for evaluating and training LLMs (Lin et al., 2021; Zhao and Anderson, 2023; Chia et al., 2024). For a detailed study, we refer readers to this comprehensive survey (Giadikiaroglou et al., 2024).

Advanced prompting techniques for reasoning. Wei et al. showed that chain-of-thought, which is a prompting technique that involves a ⟨input, chain of thought, output⟩ triple, enhances performance for reasoning based tasks. Chain-of-thought replaces a typical example answer with a more detailed step by step logical sequence. This results in stronger performance related to commonsense and symbolic reasoning. In the symbolic reasoning category, they also demonstrated that chain-of-thought improved performance for a simple state tracking task. Despite its success, CoT requires hand-designed in-context examples which are expensive and hard to scale extensively. Recently proposed Tree-of-Thought (ToT) (Long, 2023) framework incorporated additional modules to automate the process as well as mimic the trail-and-error reasoning process of humans. ToT significantly improved the performance over CoT on Sudoku

Puzzle solving tasks.

3 Method

3.1 LLM solver for Wordle

Our system for benchmarking the performance of open-source LLMs on Wordle puzzles involves the development and integration of several key components. Each component plays a critical role in creating a robust system capable of evaluating different models effectively.

Wordle Game Logic. The foundation of our system is the replication of the original Wordle game logic. This component involves implementing the core rules:

- Selecting a target word from a predefined list.
- Providing feedback for each guess, indicating correct letters in the correct position, correct letters in the incorrect position, and incorrect letters.
- Enforcing the constraints of the game, such as the maximum number of allowed guesses.

HuggingFace Model Prompting Framework. To ensure consistent and effective communication with various LLMs, we develop a flexible prompting framework. This component includes:

- Creating a standardized system prompt that instruct the models on how to play Wordle.
- Developing a unified chat interface that can be used across different models available on HuggingFace.

Interaction and Metrics Processing. This component is crucial for capturing and evaluating the models' performance. It involves:

- Establishing an interactive play, where the models generate sequential guesses for the Wordle puzzle.
- Developing algorithms to process the conversation histories, recording each guess and the feedback received.
- Implementing metrics to evaluate the performance of the models. These metrics include basic performance metrics and learning-from-mistake metrics.

With all the above components ready, the following procedure outlines how to use our system to evaluate different LLMs:

1. Initialize the Wordle Game: Start by setting up the Wordle game logic, including selecting a target word.
2. Prompt the Model: Use the developed prompting framework to instruct the selected LLM on playing Wordle.
3. Capture and Process Guesses: Record the guesses made by the model and the feedback provided by the game logic.
4. Calculate Metrics: Process the recorded interactions to calculate the defined performance metrics.
5. Analyze Results: Use the calculated metrics to analyze the performance of the model, identifying strengths and areas for improvement.

By integrating these components and following the outlined procedure, we can systematically benchmark the performance of various open-source LLMs on Wordle puzzles.

3.2 Prompting

The initial prompt is the instruction of the model to play Wordle games. The instruction explains the rules of the Wordle game, and lists the three different possibilities for each letter in a given 5 letter word guess.

In our framework, a Wordle game consists of multiple "turns", where each turn is a pair of guesses and feedback. The feedback, which is automated, returns information about the validity of each of the 5 letters in the word. For example, it might say "Letter "A" at Position 1 is in the wrong position but is in the word", and similarly for the other remaining 4 letters. This guess and feedback exchange will happen until the model guesses the word correctly or until it exhausts the maximum number of guesses.

The prompt for zero-shot learning would only give the instructions before a Wordle game begins. For k-shot learning, we instead provide the model with k example games. In these example games, we selected some example sequence of guesses that a human might make. If the answer word is "THREE", a human might make the guesses "BAGEL", "COVER", and then "THREE".

We then provide these guesses in the example games, but with explanations for the choice of guess words. The explanation contains a list of previously guessed words (history), a list of valid letters, and a list of invalid letters (letters known to not be in the word). Specifically, the format is:

```
"guess: "{guess}",

"reasoning" :
"The list of previously guessed words are: {history}
The list of valid letters are: {valid_letters}
The list of invalid letters are: {invalid_letters}

The previous guess was: {prev_guess}
Given the previous feedback, we know that
1. The letter at position 1 ({prev_guess[0]})
        was {position_info[0]}
2. The letter at position 2 ({prev_guess[1]})
        was {position_info[1]}
3. The letter at position 3 ({prev_guess[2]})
        was {position_info[2]}
4. The letter at position 4 ({prev_guess[3]})
        was {position_info[3]}
5. The letter at position 5 ({prev_guess[4]})
        was {position_info[4]}

Improving on this feedback,
the new guess is {guess} "
```

Position_info[i] states whether the i-th letter is in the word and in the correct position. Ideally, the model copies this type of elaborated reasoning to make informed guesses, and includes its reasoning as part of the output. In our early experiments, we found that language models typically have a difficult time keeping track of positions of letters, remembering its history of guesses, and the validity of letters. Because language models tend to make these kind of mistakes, we explicitly included these details for our example guesses.

4 Experiments

Outline. We first describe our experimental settings in Section 4.1. Then, we introduce our evaluation metrics in Section 4.2. Our experimental results are given in Section 4.3, followed by our discussion in Section 4.4.

4.1 Settings

Models. We use two popular and state-of-the-art open-sourced LLMs including Meta's LLama3 8B[3] and Zephyr 7B β[4] from HuggingFace. These two models are powerful yet small enough to fit into our computational resources. Llama3 is the latest generation in a series of advanced large language models developed by Meta (Touvron et al., 2023). LLama3 8B is the smallest introduced version, which have shown significant improvements in performance and capabilities compared to their predecessors. Zephyr is a series of language models designed to function as highly effective assistants. The Zephyr 7B β model, the second in this series, is a fine-tuned version of Mistral 7B (Jiang et al., 2023). It has been trained using a combination of publicly available and synthetic datasets through the method of Direct Preference Optimization (DPO) (Rafailov et al., 2024), enhancing its ability to understand and respond to a wide range of tasks with improved accuracy and helpfulness.

Datasets. We used a list of 488 common 5 letter words from https://byjus.com/english/5-letter-words/. While there is a list of all of the words used in the official Wordle game, it contained some words that were too rare for a language model to consider.

Baselines. We compare our proposed few-shot prompt with zero-shot prompt.

Computational resources. All experiments are run on a Google Cloud Platform (GCP) virtual machine with a 26GB NVIDIA T4 GPU. We sincerely thank Prof. Nanyun Peng for providing us the GCP credits.

Other implementation details. For loading pretrained LLMs and generating answers, we use the transformers library[5] from HuggingFace. Due to the computational constraints, we leverage the 4-bit quantization of the bitsandbytes[6] library to load and run inference on all models. We also use the vLLM library (Kwon et al., 2023) to boost the inference speed.

4.2 Evaluation metrics

We use two sets of metrics, basic performance and learning from mistake, to evaluate the performance of LLMs on Wordle.

Basic performance. This criteria includes some existing metrics that have already been commonly used in other systems and in actual game plays.

- The number of guesses required to find the correct answer. The value ranges from 1 to 5 with a smaller number indicating better performance.
- With each guess, how many letters are correct (*correct_letter_pct*) and how many letters

[3] https://huggingface.co/meta-llama/Meta-Llama-3-8B
[4] https://huggingface.co/HuggingFaceH4/zephyr-7b-beta
[5] https://huggingface.co/docs/transformers/en/index
[6] https://github.com/TimDettmers/bitsandbytes

Model	correct_pct (%, ↑)	correct_letter_pct (%, ↑)	end_letter_acc (↑)	success_pct (%, ↑)	avg_guess (↓)
LLama3 8B	**23.5**	**38.1**	**1.6**	**7.2**	**3.9**
Zephyr 7B β	10.3	25.1	0.6	0.0	-

Table 1: Comparing the basic performance of the zero-shot prompting between LLama3 8B and Zephyr 7B β. The number of Wordle games is fixed to 488.

Number of shots	correct_pct (%, ↑)	correct_letter_pct (%, ↑)	end_letter_acc (↑)	success_pct (%, ↑)	avg_guess (↓)
0	**23.5**	**38.1**	**1.6**	**7.2**	3.9
1	20.9	35.9	1.4	1.8	3.8
2	<u>21.0</u>	<u>36.3</u>	<u>1.5</u>	2.0	4.1
3	20.8	34.5	1.4	3.1	3.9
4	20.2	35.1	1.4	<u>3.5</u>	**3.4**
5	20.1	35.7	1.4	2.9	<u>3.6</u>

Table 2: The basic performance of LLama3 8B when changing the number of in-context examples. The best performance is marked in bold while the second best is underlined. The number of Wordle games is fixed to 488.

Model	reuse_wrong_letters (%, ↓)	reuse_wrong_positions (%, ↓)	undo_correct_letters (%, ↓)
LLama3 8B	**13.0** (-21.4)	15.5 (+1.5)	**33.4** (-53.6)
Zephyr 7B β	47.3 (+12.9)	**11.6** (-2.4)	71.7 (-12.0)

Table 3: Comparing the ability to follow feedback between LLama3 8B and Zephyr 7B βin zero-shot learning. The number in parenthesis indicate the improvement compared to random guessing (i.e., given no feedback). The number of Wordle games is fixed to 488.

are correct and in the correct position (*correct_pct*)? The value ranges from 0% to 100% with a larger number indicating better performance.

- In the fifth or final guess how many letters are correct and positioned accurately (*end_letter_acc*). The value ranges from 0 to 5 with a larger number indicating better performance.
- The number of words that the model guesses correctly (*success_pct*). The value ranges from 0% to 100% with a larger number indicating better performance.
- When the model guesses the word correctly, how many turns does it need in average (*avg_guess*). The value ranges from 1 to 5 with a smaller number indicating better performance.

Learning from mistake. The basic metrics, however, do not fully capture the reasoning ability of LLMs, their ability to follow instructions and improve from feedback. In the Wordle game plays, humans use the feedback from the game to keep/reorder/replace the green/yellow/gray letters. Therefore, if having the reasoning ability, LLMs should have the same strategy. To measure this capability, we propose three new metrics to verify whether or not LLMs repeat previous mistakes including

- Reuse letters already confirmed as not present in the word (*reuse_wrong_letters*). The value ranges from 0% to 100% with a smaller number indicating better performance.
- Place a letter in a position previously determined to be incorrect (*reuse_wrong_positions*). The value ranges from 0% to 100% with a smaller number indicating better performance.
- Fail to utilize known correct letters and their confirmed positions (*undo_correct_letters*). The value ranges from 0% to 100% with a smaller number indicating better performance.

4.3 Results

Comparing different LLMs. Table 1 shows that LLama3 8B outperforms Zephyr 7B β in all 5 basic metrics. This result is expected because LLama3 8B is released after Zephyr 7B β, having more parameters and a larger training dataset. Surprisingly, Zephyr 7B βdoes not guess a single correct word (within 5 guesses) in 488 different games that it plays. Interestingly, some of the words guessed are not valid English words (e.g. LEALP) even though our prompts explicitly tell the model to guess a valid 5-letter English word.

The effect of the number of shots. As can be seen in Table 2, zero-shot learning outperforms in-context learning in 4 out of 5 basic metrics. We do not observe any specific patterns in the basic performance when increasing the number of demonstrations.

Can LLMs learn from mistake? Table 3 indicates that all models use the correct letter in the right place well and much better than random guessing. However, LLama3 8Buses the wrong position information poorly, even slightly worse than random guessing. LLama3 8B does not reuse invalid letters frequently while Zephyr 7B β does reuse invalid letters quite often.

4.4 Discussion

The performance of open-source LLMs on Wordle puzzles has been generally disappointing. A key concern is model size, as the underperformance could be attributed to using models with fewer than 8 billion parameters and the usage of 4-bit quantization. Computational resource constraints limited the deployment of larger, more capable models, which might have performed better. Surprisingly, the models exhibited difficulties in following basic instructions and processing feedback effectively, with LLama3 8B showing some improvement over others but still falling short of expectations. Several potential factors may contribute to these issues, including the impact of subword tokenization and inherent limitations in creatively exploring a broad range of possibilities. Additionally, the log probabilities across 5-letter words were uniformly low, indicating a potential systemic issue in how these models handle word-level tasks in games like Wordle.

5 Conclusion

In this project, we develop a chat system and introduce automatic evaluation metrics to benchmark the performance of open-source LLMs on the Wordle puzzle. Our results indicated that open-source LLMs, much like ChatGPT, struggle with Wordle, often failing to learn from their mistakes and barely producing correct guesses. These findings underscore the need for further refinement and optimization of LLMs for specialized tasks, contributing to a broader understanding of their capabilities and limitations. A potential direction for our future works is to build a Wordle game dataset and finetune models on it to improve the performance of large language models in playing games such as Wordle.

6 Work Distribution

Jinwoo Baik: Wrote the initial code to emulate the Wordle game, and also wrote the initial code for metrics that would measure the model's performance. Afterwards, contributed code for prompting and the pipeline for the experiments. Ran several experiments with various parameters, and analyzed their results. Prepared materials for the presentation, and wrote parts of Related Works and Method sections.

Dang Nguyen: Fixed the Wordle game logic and contributed to the initial prompting notebook. Implemented the first version of learning-from-mistake metrics. Analyzed the experimental results and derived some conclusions. Prepared the Results part in the final presentation. Drafted the following sections in the final report: Abstract, Introduction, Related works (Solving Wordle using LLMs and LLMs for solving other puzzles), LLM solver for Wordle, Experiments, and Conclusion.

Lucas Bandarkar: Contributed to the Wordle game logic and built an object-oriented version of it. Unsuccessfully set up a Google Cloud platform and the credits. Developed the HuggingFace code to prompt the model and interact with the Wordle game object. This involved setting up end-to-end environment in which the game was created, prompted to the model in a full conversation, and concluded. Iterated through some prompts to determine what worked and what didn't. Developed the code for the metrics and the subsequent analysis, including running several of the experiments presented in class. Finally, organized and built the presentation.

Sara Khosravi: Contributed to implement few shot prompts and the corresponding unit tests for the game. Also added corresponding instructions in the Github repository readme format. Run prompts and instructions were further changed by other collaborators. Unsuccessful try for changing the python and Jupyter Notebook files to python scripts. Contributed to prepare presentation slides and its design and template. Added reinforcement learning in solving wordle game to the literature review of the paper (revised and omitted later). Set up Google Cloud service. In person contribution to wordle game logic design.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yew Ken Chia, Vernon Toh Yan Han, Deepanway Ghosal, Lidong Bing, and Soujanya Poria. 2024. Puzzlevqa: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns. *arXiv preprint arXiv:2403.13315*.

Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. Puzzle solving using reasoning of large language models: A survey. *arXiv preprint arXiv:2402.11291*.

Akshat Gupta. 2023. Are chatgpt and gpt-4 good poker players?–a pre-flop analysis. *arXiv preprint arXiv:2308.12466*.

Chenghao Huang, Yanbo Cao, Yinlong Wen, Tao Zhou, and Yanru Zhang. 2024. Pokergpt: An end-to-end lightweight solver for multi-player texas hold'em via large language model. *arXiv preprint arXiv:2401.06781*.

Adam Ishay, Zhun Yang, and Joohyung Lee. 2023. Leveraging large language models to generate answer set programs. *arXiv preprint arXiv:2307.07699*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Yinghao Li, Haorui Wang, and Chao Zhang. 2023. Assessing logical puzzle solving in large language models: Insights from a minesweeper case study. *arXiv preprint arXiv:2311.07387*.

Bill Yuchen Lin, Ziyi Wu, Yichi Yang, Dong-Ho Lee, and Xiang Ren. 2021. Riddlesense: Reasoning about riddle questions featuring linguistic creativity and commonsense knowledge. *arXiv preprint arXiv:2101.00376*.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017.

Jieyi Long. 2023. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*.

Michael G. Madden. 2024. Chatgpt struggles with wordle puzzles, which says a lot about how it works.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

David Noever and Ryerson Burdick. 2021. Puzzle solving without search or human knowledge: An unnatural language approach. *arXiv preprint arXiv:2109.02797*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Jim Thompson. 2024. Wordle solver virtual assistant (wsva) testbed. https://github.com/jimthompson5802/wordle_solver.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023.

Exploring large language models for communication games: An empirical study on werewolf. *arXiv preprint arXiv:2309.04658*.

Husain Zaidi. 2023. Llm-powered wordle-solver agent. https://github.com/husainhz7/wordle-llm-solver.

Jingmiao Zhao and Carolyn Jane Anderson. 2023. Solving and generating npr sunday puzzles with large language models. *arXiv preprint arXiv:2306.12255*.