Minimax Tree of Thoughts: Playing Two-Player Zero-Sum Sequential Games with Large Language Models

Wei Guo [*1] **Xiaotian Hao** [*1] **Jianye Hao** [1] **Yan Zheng** [1]

Abstract

Large language models are being used to solve an increasing number of tasks, but the existing methods based on large language models are still not good enough in playing two-player zero-sum sequential games. In order to solve the related challenges of large language models playing two-player zero-sum sequential games, we propose Minimax Tree of Thoughts, which combines the idea of Tree of Thoughts and minimax search. Experiment results show that our Minimax Tree of Thoughts method significantly outperforms the original Tree of Thoughts method in two-player zero-sum sequential games tasks such as Word chain and game of Ghost.

1. Introduction

Large language models (LLMs) such as GPT-4 (OpenAI, 2023) are language models with a large number of parameters. In recent years, with the proposal of a series of LLM-based planning methods such as Chain of Thoughts (CoT) (Wei et al., 2023), Self-Consistency Chain of Thoughts (CoT-SC) (Wang et al., 2023), Tree of Thoughts (ToT) (Yao et al., 2023), LLMs have been discovered to have the potential to complete a wide range of tasks, such as games, common sense reasoning, math problems (Yao et al., 2023; Wang et al., 2023; Wei et al., 2023), etc. However, in practice, although existing LLM-based planning methods such as ToT are capable of single-player decision-making tasks such as game of 24, mini crosswords game (Yao et al., 2023), they do not perform well in two-player zero-sum sequential games. Two-player zero-sum sequential games are a huge category of tasks, which includes chess, Tic Tac Toe, etc. In this paper, we only consider the games with perfect information. The reason why methods like ToT are not good at playing two-player zero-sum games is that these methods only try to plan a trajectory that leads to win, but they do not sufficiently consider the adversarial behavior of the opponent, which is necessary for playing two-player zero-sum sequential games at a high level.

To design a LLM-based method that can play two-player zero-sum sequential games effectively, we borrow the idea from minimax search (Knuth & Moore, 1975). Minimax search is a typical method for solving two-player zero-sum sequential games without LLM. The original minimax search algorithm suffers from the problem of high computational cost. To deal with that, various methods have been introduced, the most typical method being alpha-beta pruning (Knuth & Moore, 1975) which can cut off some sub-optimal branches by recording previous search results. Although alpha-beta pruning can improve the efficiency of minimax search, its improvement is not stable. In particular, there's no improvement in the worst case (Knuth & Moore, 1975). To deal with the problem of too many search branches in minimax search, we borrowed the idea from ToT and proposed the minimax tree of thoughts (Minimax ToT), which uses LLMs to generate only a small set of search branches thus significantly enhance the performance. Moreover, we further improve the efficiency of minimax search by limiting the number of search layers and using Monte Carlo rollout (Tesauro & Galperin, 1996) to estimate the value of the bottom nodes.

We compared our method with ToT on two tasks, Word chain (Wise & Forrest, 2003) and Ghost (Morehead et al., 2001). In Word chain, our method won about 71% of the games against ToT. In Ghost, our method won 81% of the games against ToT. In addition, we also conducted ablation experiments on the two parts of our method, minimax search and rollout, proving that both parts are helpful in improving performance.

2. Background

The **two-player zero-sum sequential game** is a sequential game played by two players in which the sum of the payoffs of two players in every outcome is always zero. The game is often represented as a game tree (Kuhn, 1950). Without loss of generality, we define the value of the game as player 1's

[*]Equal contribution [1]College of Intelligence and Computing, Tianjin University, Tianjin, China.

Workshop on Large Language Models and Cognition at the 41st International Conference on Machine Learning, Vienna, Austria, 2024.

pay off. Therefore, player 1, who is trying to maximize the value of the game, is called the maximizing player, while player 2, who is trying to minimize the value of the game, is called the minimizing player.

Tree of Thoughts (ToT) is a LLM-based planning method that use the idea of tree search. ToT has two variants, ToT-BFS and ToT-DFS, which corresponds to breads-first search and depth-first search, respectively. Both of them are using a LLM-based thought generator to generate actions, and using a LLM-based states evaluator to evaluate states. The thought generator can generate actions through two different strategies, namely sample and propose. The states evaluator evaluates new states by either valuing or voting. The goal of ToT is to obtain a plan through tree search. However, ToT does not perform well in two-player zero-sum games. This is because ToT does not assume the optimality of the opponent. Therefore, when a ToT-based agent confronts an intelligent opponent, the actual trajectory may deviate from the trajectory planned by the ToT, which makes the plan obtained by ToT no longer practical.

The **Minimax search** is a decision-making method for two-player zero-sum sequential games, which was widely used in board games AI. The Minimax search algorithm simulates the adversarial strategies of both players by taking the maximum and minimum operations during the tree search process, thereby obtaining the optimal decision for each players. However, in the original minimax search, each player loop over all possible actions, which result in large computational cost. Although researchers later proposed many ways to reduce its cost, such as alpha-beta pruning. However, the effect of alpha-beta pruning is unstable and does not guarantee any improvement in the worst case.

3. Minimax Tree of Thoughts: Playing Two-Player Zero-Sum Sequential Games with LLMs

To improve the respective drawbacks of ToT and minimax search, we propose the Minimax Tree of Thoughts (Minimax ToT). In Minimax ToT, we adopt the idea from ToT to use LLMs to generate actions. This allows us to take advantage of the reasoning capabilities of the LLMs to generate reasonable actions. Moreover, we borrow the idea from minimax search of taking max and min operations during searching to make our method adapt to two-player zero-sum sequential games. Since the time complexity of the minimax search increases exponentially with the search depth, we introduced a maximum search depth D to limit.

The implementation of the Minimax ToT involves answering three fundamental questions, which are: 1. How to generate actions, 2. How to limit the depth of the search, and 3. How to reduce the LLM calls.

1. How to generate actions. We generate actions in three steps: generating, filtering, and sorting. These three steps are similar to the thought generator of ToT.

- **Generationg**: At each sate, Minimax ToT generates at most n actions (where n is a preset constant) for each game state node. Similar to ToT, actions are generated by either multiple or single samplings from the LLM. Unlike ToT, we have removed the restrictions of i.i.d during sampling. This allows us to reduce the generation of repeated actions by prohibiting actions already sampled through prompting.

- **Filtering**: Minimax ToT then filters the generated actions to refine the actions. The filtering process follows the principles below:

  1. Remove invalid actions (e.g. actions that breaking the constraints of the task)
  2. Remove duplicate actions.
  3. Remove dominated actions (optional).

- **Sorting**: Placing the better actions to the front throught sorting. The criteria of the sorting can be provided either by LLM or by hand-crafted heuristics. This step can enhance the effect of alpha-beta pruning.

2. How to limit the depth of the search. Since the time complexity of the minimax search algorithm grows exponentially with the increase in search layers, we limit the search to at most D layers, where D is a preset constant. If the algorithm does not reach an endgame after the D-th layer, it needs to estimate the value of the D-th layer nodes. The estimation is done through rollouts. The algorithm will randomly rollout multiple trajectories of the game, and then take the average of the end-game values from different trajectories as the estimation of the node's value. During rollout, the generation of actions still uses the LLM, but only sample one action each time.

3. How to reduce LLM calls. In experiments, we observed that there were numerous calls to LLMs with exactly same prompt. In order to reused the previous result of LLM calls hence to reduce the usage of LLM tokens, we use a dictionary to cache the LLM output to reused them later. The experiment shows that this trick can reduce the number of LLM calls effectively, thereby saving the tokens.

Whenever a player makes a decision, the Minimax ToT will go through 3 steps. Step 1, generating a list of actions using a LLM. Step 2, valuing each actions through minimax search and rollout by calling the MinimaxToT function. Step 3, selecting the action with the highest value to execute. If there are multiple actions with same value, a random break tie is performed. The main content of the algorithm

Algorithm 1 MinimaxToT

```
Function MinimaxToT(state, depth, α, β)
if GameEnds then return EndGameValue(state)
if depth = 0 then
    Call RollOut(state) for n times and return the average.
end if
actionList ← GenerateAternativeActions(state)
if is maximizing player then
    maxEval ← −∞
    for action in actionList do
        eval ← MinimaxToT(NextState, depth - 1, α, β)
        maxEval ← max(maxEval, eval)
        α ← max(α, eval)
        if β ≤ α then break
    end for
    return maxEval
else
    minEval ← ∞
    for action in actionList do
        eval ← MinimaxToT(NextState, depth - 1, α, β)
        minEval ← min(minEval, eval)
        β ← min(β, eval)
        if β ≤ α then break
    end for
    return minEval
end if
```

Table 1. Comparation between Minimax ToT and ToT.

TASK	MINIMAX TOT WINNING RATE
WORD CHAIN	$\approx 71.15\%$
GHOST	81%

in implemented in the function MinimaxToT. Algorithm 1 is the pseudo-code of MinimaxToT function.

The underlying LLM of the Minimax ToT algorithm can be chosen flexibly such as GPT-4, Llama 2 (Touvron et al., 2023). In conclusion, the Minimax ToT provides an approach to integrate minimax search with LLMs, and future work can be further improved based on this approach.

4. Experiments

We tested our Minimax ToT algorithm under two tasks, namely Word chain and Ghost. The experiments show that Minimax ToT outperforms ToT in both games (See Table 1). In addition, we also conducted ablation studies between the two components of Minimax ToT, which are minimax search and rollout.

Table 2. Ablation studies of the two components of Minimax ToT on Word chain task. (WR short for Winning Rate)

PLAYER 1	PLAYER 2	WR. OF PLAYER 1
MINIMAX TOT	NO ROLLOUT	$\approx 65.38\%$
MINIMAX TOT	NO MINIMAX SEARCH	$\approx 69.23\%$

4.1. Word chain

Word chain is a popular word game in which players find words that begin with the letter that the previous word ended with, and players are not allowed to say words that are duplicates of existing words. Since LLMs are more proficient at memorizing obscure words compared to human players, the game rules that apply to human players are not fully applicable to LLMs. Generally, the LLM-based methods may encounter three types of problems when playing the original Word chain: 1) The game won't end after a long period of time; 2) The first-mover advantage is too large; 3) The game lacks diversity. To deal with these problems, we used a specific set of game rules (See Appendix A).

In the experiment, we had one agent based on the Minimax ToT act as player 1, another agent based on the ToT act as player 2. We tested a total of 52 games, which included two games for each of the 26 English letters as the first letter of the first word. In these two games, Player 1 went first in the first game, and Player 2 went first in the second game, thus covering all possible opening scenarios.

In the Minimax ToT used by player 1, we set the maximum search depth D to 3, and the rollout number for valuing the bottom nodes of the search tree to 3. The LLM we used is a GPTQ quantized (Frantar et al., 2023) version of Llama 2 7b chat model with the temperature set to 0.5. We designed few-shot examples (Brown et al., 2020) to prompt the LLM only output common words to avoid outputting words that not included in the dictionary (for detailed prompt, see Appendix B).

In the ToT method used by player 2, we used ToT-DFS to find a trajectory from current state that could lead player 2 to victory. If such a trajectory was found, player 2 would perform the first action in the trajectory; otherwise, player 2 would perform the default action (output an empty string).

The experimental results are shown in Table 1. It can be seen that the Minimax ToT agent wins 71% of the games versus the ToT agent.

In addition to the comparison with ToT, we also conducted an ablation experiments between the Minimax search and rollout modules in the Minimax ToT. The experimental setup remains the same, using each of the 26 letters as the first letter of the first word, with each letter playing a round

with Player 1 going first and Player 2 going first, for a total of 52 rounds. The experimental results are shown in Table 2. It can be seen from Table 2 that both the minimax search and rollout contribute to performance improvement.

4.2. Ghost

Ghost is a popular word game played by two players. During the game, the two players take turns to say a letter. Each player must ensure that the letter they say, combined with the existing letters, must form a prefix of a word, but cannot form a complete word (for example, "answ" is a prefix of "answer", but it does not constitute a word itself). During each turn, a player can either add a letter or challenge the previous player to prove that the present fragment initiates a word. If the challenged player can provide such a word, the challenger loses; if not, the challenged player loses. Typically, in a Ghost game, the winner isn't decided by a single round. For simplicity, we adopted a single-round decisive rule. Also, to avoid early game-ends due to abundant short words in English, a word length limit is usually applied. In our experiment, the words must have at least 5 letters.

We tested Minimax ToT and ToT in Ghost task. In our experiment, player 1 is a Minimax ToT-based agent, while player 2 is a ToT-based agent. We let player 1 and player 2 compete 100 games, and the first move in each game was randomly selected. Player 1 used a quantified version of Llama2 13b chat model, with the temperature set to 0.5, the maximum search layer D set to 3, and the rollout number for valuing the bottom nodes set to 3. For player 2, we used ToT-DFS to find a plan that would lead to a win for player 2 starting from the current state. If such a plan could be found, player 2 would execute the first action of the plan; otherwise, player 2 would execute the default action (challenge player 1). We used three free dictionaries, NLTK (Bird et al., 2009), WordNet, and Educalingo to verify whether a word exists.

The experimental results are shown in Table 1. Our Minimax ToT agent wins 81% of the games versus ToT agent.

Finally, to verify that our method is also applicable to other LLMs besides Llama 2, we tested a 20-round match between Minimax ToT and ToT on GPT 3.5 Turbo, and the Minimax ToT wins 70% of the games. This proves that our method is applicable to different foundation models.

5. Related works

5.1. LLM-based Planning methods

(Huang et al., 2022) suggested that the pretrained LLM can act as a zero-shot planner. The typical multi-step LLM-based planning method is Chain of Thought (CoT), which uses few shot in-context examples to enable LLMs to plan step-by-step. Zero-shot-CoT (Kojima et al., 2023) achieves the similar result by adding something like "Let's think step by step." in the prompt. Self-consistent CoT (CoT-SC) plans multiple trajectories by running CoT multiple times, then output the answer with the highest frequency. Tree of Thoughts (ToT) is a tree search methods in which thoughts are generated by LLMs. In A recently published LLM-based tree search planning method is RAP, which employs a world model to simulate the value of different plans through Monte Carlo Tree Search (MCTS) (Kocsis & Szepesvári, 2006). However, these works are not focused on two-player zero-sum sequential games.

5.2. Methods for playing two-player zero-sum sequential games

A typical method for playing two-player zero-sum sequential games is minimax search, which can be regarded as a special case of backward induction (Watson, 2002). The minimax search with alpha-beta pruning has achieved superhuman performance in board games such as chess (Campbell et al., 2002) and checkers (Schaeffer et al., 1992). Another method for two-player zero-sum sequential games is using Monte Carlo tree search (MCTS), which borrowed ideas from bandit problems. In 2016, AlphaGo (Silver et al., 2016) achieved superhuman level in playing the game of Go. In 2019, AlphaStar (Vinyals et al., 2019) was proposed, which was designed to the game of Starcraft. However, these works are not LLM-based.

6. Discussion

6.1. Limitations

The limitations of our algorithm are mainly on computational cost. The cost of both minimax search and Monte Carlo rollout are too high when the game tree is large. Although we have employed several tricks to limit the computational cost, such as alpha-beta pruning and limiting the search depth D, but the computational cost is still depend on the length of trajectory.

6.2. Future directions

In the future, we'll try to eliminate the limitations mentioned above. There are several ways to do that, such as employing MCTS with UCT (Kocsis & Szepesvári, 2006) or pUCT (Rosin, 2011).

6.3. Conclusion

This paper presents a method combining Minimax search with LLM, and verifies the feasibility of this method, which provides a foundation for the subsequent work.

7. Acknowledgements

We extend our sincere gratitude to the reviewers for their insightful feedbacks. We also thank Tianjin University for providing necessary facilities and resources. Finally, we wish to express our appreciation to our families and friends for their continuous support and encouragement.

References

Bird, S., Klein, E., and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Campbell, M., Hoane, A., and hsiung Hsu, F. Deep blue. *Artificial Intelligence*, 134(1):57–83, 2002. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(01)00129-1.

Educalingo. About educalingo, 2024. URL https://educalingo.com/en/about.

Fellbaum, C. (ed.). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA, 1998. ISBN 978-0-262-06197-1.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *CoRR*, abs/2201.07207, 2022. URL https://arxiv.org/abs/2201.07207.

Knuth, D. E. and Moore, R. W. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(75)90019-3.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M. (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners, 2023.

Kuhn, H. W. Extensive games. *Proceedings of the National Academy of Sciences of the United States of America*, 36 (10):570–576, 1950. ISSN 00278424.

Morehead, A., Mott-Smith, G., and Morehead, P. *Hoyle's Rules of Games*. Penguin Publishing Group, 2001. ISBN 9781101100233.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

Princeton University. About wordnet, 2010. URL https://wordnet.princeton.edu/.

Rosin, C. D. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3): 203–230, Mar 2011. ISSN 1573-7470. doi: 10.1007/s10472-011-9258-6.

Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., and Szafron, D. A world championship caliber checkers program. *Artificial Intelligence*, 53(2):273–289, 1992. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(92)90074-8.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. ISSN 1476-4687. doi: 10.1038/nature16961.

Tesauro, G. and Galperin, G. On-line policy improvement using monte-carlo search. In Mozer, M., Jordan, M., and Petsche, T. (eds.), *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, Nov 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1724-z.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *ICLR 2023*, 2023.

Watson, J. *Strategy: An Introduction to Game Theory*. W.W. Norton, 2002. ISBN 9780393976489.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Wise, D. and Forrest, S. *Great Big Book of Children's Games*. SPANISH IMPORTS - BGR. McGraw-Hill Education, 2003. ISBN 9780071422468.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models, 2023.

A. The set of Word chain rules used in experiments

We have mentioned that when LLM-based agent playing the original Word chain, there are three problems, which are: 1) The game won't end after a long period of time; 2) The first-mover advantage is too large; 3) The game lacks diversity. To deal with these problems, we used a specific set of game rules. Next, we will introduce the cause of these problems and the rules we used to solve these problems.

- **The game won't end after a long period of time** mainly manifests in that LLMs are too good at memorizing words. Therefore, the original version of Word chain is too easy for LLMs, which can easily lead to very long game trajectories. Therefore, in order to determine a winner within limited number of steps, we have borrowed a rule from a frequently played variant of Word chain, that is, the player whose words' total length exceeding a certain threshold wins the game.
- **The first-move advantage being too large** can be caused by many reasons. For example, during the experiment, we found that after adopting the rule of determining the winner by the total word length exceeding a threshold, when we prompt the LLM to output as long words as possible, it would output very long names of chemical substances or long place names. The length of these rare nouns usually far exceeds the threshold we set for winning the game, which leads to the first player always winning the game. Moreover, even if only shorter words are used (for example, only allowing the LLM to use common adjectives), when both sides adopt the same algorithm (such as the ToT), the winner is also likely to be the first player, because the first player acts first, so it is easier to exceed the threshold of the total letters first. To limit the first-move advantage, we first limited the range of legal words to exclude nouns, thus excluding those particularly long place names and names of chemical substances. In addition, we also added a rule that only half of the letter count of the first word of the first player is calculated (rounded down when it cannot be evenly divided) to eliminate the first-move advantage.
- **The game lacks diversity** is another problem to consider. It happens when we ask LLMs to output long verbs, we will get a lot of words ending with letter e. It seems that verbs are very likely to have e as its end letter. As a result, the game trajectories are similar between rounds. To deal with this problem, we limited the valid word set in the game to adjectives. Experiments showed that this restriction effectively avoided the previous situation where every round was quite similar.

Finally, the detailed rules of our Word chain variant is as follows:

- Two players take turns coming up with words, and the first letter of each word said by a player must be the last letter of the word said by the previous player.
- The first-move player and the first letter of the first word raised by the first-move player differ from game to game, which can either be randomly chosen or predetermined.
- The valid words is limited to adjectives.
- Whoever fails to respond or responds incorrectly loses the game. The words are verified through WordNet (Fellbaum, 1998; Princeton University, 2010) and Educalingo (Educalingo, 2024), two free online dictionaries, to confirm their existence and correct part of speech. If both players can respond to each other's words, the first to reach 40 letters wins.
- To reduce the first-move advantage, only half of the letters in the first word raised by the first player are counted (if it's not an even number, round down).

B. LLM prompts

B.1. Word chain

In the task of Word chain, we use the following promt to generate actions:

```
Give me 10 base form nouns starting with "a". The longer the words the better. Make sure
    the words are commonly used. Don't output compound words with hyphens.
1. Application; 2. Association; 3. Architecture; 4. Argumentation; 5. Abstraction; 6.
    Advancement; 7. Anticipation; 8. Appreciation; 9. Authorization; 10. Accommodation
```

```
Give me 10 base form verbs starting with "c" excluding "come, alienate, exam". The longer
    the words the better. Make sure the words are commonly used. Don't output compound
    words with hyphens.
1. Communicate; 2. Collaborate; 3. Contribute; 4. Calculate; 5. Coordinate; 6. Construct;
    7. Criticize; 8. Cultivate; 9. Celebrate; 10. Customize
Give me 10 base form adjectives starting with "{letter}"{excluding_prompt}. The longer the
     words the better. Make sure the words are commonly used. Don't output compound words
    with hyphens.
```

where `{letter}` is then replaced with the starting letter needed to satisfy, the `{excluding_prompt}` is then replaced with a list of previous words to avoid. Note that we have limited the word to be commonly used and without hyphens, since otherwise the word output by LLM may not included in a dictionary.

After obtaining the action list, we filter the actions using the following LLM prompt:

```
Is "bigger" an adjective?
YES
Is "heavy" an adjective?
YES
Is "make" an adjective?
NO
Is "{word}" an adjective?
```

where the `{word}` is then replaced with some word in the action list. This prompt aims to verify whether an action is valid.

B.2. Game of Ghost

In the task of Ghost, we use the following prompt to generate actions:

```
Please provide a word that starts with the letters "exc" but not starts with "excl" or "
    excu" and has a length of at least 5.
exclaim
Please provide a word that starts with the letters "a" has a length of at least 6.
anyone
Please provide a word that starts with the letters "can" has a length of at least 5.
candidate
Please provide a word that starts with the letters "{prefix}" but not starts with "{
    bannedPrefix}" and has a length of at least {length_min}
```

where the `{prefix}` is then replaced with current word fragment, `{bannedPrefix}` is then replaced with a list of prefixes that the word shouldn't start with, the `{length_min}` is then replaced with the minimum length of required.

Note that the above prompt is a little subtle. As an illustrative example, let's assume the current fragment is "stam". We may prompt the LLM to output a word that starts with "stam" and has at least 6 letters. This will prevent the LLM from outputting a word with exactly 6 letters, such as "stamp", since according to the rules of Ghost, each player must make sure the letter they add to the fragment does not complete a word. If the LLM outputs "stamina", we will take the fifth letter "i" and add it to the action list. In the next sample, we may instruct the LLM to output a word that does not start with "stami" to avoid sampling duplicate actions.

After obtaining the action list, we filter the actions using the following LLM prompt:

```
Is "apple" a correct English word?
YES
Is "narrat" a correct English word?
NO
Is "like" a correct English word?
YES
IS "{word}" a correct English word?
```

where `{word}` is then replaced with the current fragment combines with the current action, this will check if an action is indeed valid.