Reinforcement Learning for Chain of Thought Reasoning: A Case Study Using Tic-Tac-Toe

Preprint · July 2024

DOI: 10.13140/RG.2.2.16291.67365

2 authors:

Chatgpt C-Lara-Instance

19 PUBLICATIONS **6** CITATIONS

SEE PROFILE

Manny Rayner

University of South Australia

245 PUBLICATIONS **1,861** CITATIONS

SEE PROFILE

Reinforcement Learning for Chain of Thought Reasoning: A Case Study Using Tic-Tac-Toe*

ChatGPT-4 C-LARA-Instance[1,2], Manny Rayner[2]

[1]OpenAI, US; [2]University of South Australia, Australia;

Abstract

"Chain of Thought" (CoT) is an increasingly popular methodology for querying Large Language Models, where the LLM is told to formulate a response by in effect thinking aloud. CoT is known to increase accuracy in many domains. A practical problem, however, is that there is generally little available data from which to extract examples to guide the CoT process. Here, we describe an experiment in which we used a reinforcement learning method to evolve useful few-shot examples for CoT in the context of the game of Tic-Tac-Toe. The experiment consisted of 40 cycles. In each cycle, the CoT player played two games each against five different players, including a random player and a perfect minimax player. The CoT protocols were then filtered, with the successful ones used in the next round. The CoT player's smoothed average cycle score improved from 5.6/10 at the beginning to 7.0/10 at the end, significant at $p = 0.01$; the smoothed move decision correctness, as evaluated by the minimax player, improved from 83% to 92%, significant at $p = 0.001$. Other techniques used include matching few-shot examples to positions, majority voting, and parallel execution of GPT-4 queries. We discuss the significance of the findings and suggest ways to apply similar methods to an ongoing project where GPT-4 is used to produce annotated multimodal texts for language learning.

*Authors in alphabetical order.

Contents

1. Introduction and overview

Chain of Thought (CoT) reasoning is an emerging approach in the realm of large language models (LLMs) that emphasizes the importance of intermediate reasoning steps in achieving more accurate and reliable outputs. This methodology leverages the ability of LLMs to "think aloud," thereby improving their capacity to handle complex tasks through explicit reasoning processes. CoT has become increasingly popular among researchers and practitioners using LLMs, as it offers a structured way to enhance the model's performance on tasks that require multi-step reasoning. People using CoT sometimes draw a parallel with the distinction between "System 1" and "System 2" thinking introduced by Kahneman's influential book *Thinking, Fast and Slow* (Kahneman, 2011). In this perspective, normal LLM prompts give responses analogous to fast, reactive System 1 thinking, while CoT prompts gives responses more similar to slow, reflective System 2 thinking. The identification is only approximate, but can be helpful for developing intuitions about CoT.

Few-shot examples are often used in both non-CoT and CoT reasoning to provide the model with specific instances illustrating how to approach a task. Thus, in a non-CoT prompt, the few-shot examples essentially say "Respond to this, using the following examples as guidelines", while in a CoT prompt the effect is more "Think about this aloud and then respond, using the following examples as guidelines". The examples help focus the model's reasoning process, making it more effective in generating the desired outputs. However, manually creating few-shot examples can be time-consuming and challenging, particularly for complex tasks. Therefore, the ability to generate CoT material automatically is highly desirable, as it would streamline the process and make it more scalable.

The success of reinforcement learning approaches, such as AlphaZero (Silver et al., 2018; Sadler and Regan, 2019), in mastering games like chess and Go suggests the idea of exploring similar methodologies for creating few-shot examples in CoT reasoning. AlphaZero's ability to learn and improve through self-play, without relying on pre-existing human knowledge, presents a compelling model. Here, our primary goal is to investigate whether we can emulate the AlphaZero methodology to develop few-shot examples for CoT, thereby enhancing the performance of LLMs in tasks requiring multi-step reasoning. Our approach is is in particular motivated by Leopold Aschenbrenner's remarks in his essay "Situational Awareness." Aschenbrenner highlights the impending "data wall" that limits the current pretraining paradigm due to the finite availability of high-quality internet data. He references AlphaZero and suggests that synthetic data and self-play methods, akin to how humans learn through iterative practice and internal monologue, could be pivotal in overcoming these data constraints and improving sample efficiency in LLMs (Aschenbrenner, 2024, pp. 26–29).

Given that Tic-Tac-Toe, though trivial for humans, is well known to be challenging for LLMs (Mollick, 2024), we considered that it would be a good starting point. In the experiments described here, we show that it is possible to apply a version of reinforcement learning to the task of developing a CoT player for Tic-Tac-Toe, creating few-shot examples from CoT protocols which led to correct game decisions. The prompts that use these few-shot examples give rise to more CoT protocols, which can potentially be reused as new few-shot examples. Over 40

cycles of game play followed by extraction of CoT protocols, we demonstrate an improvement in the CoT player's performance, measured both in terms of higher game scores against an ensemble of five baseline Tic-Tac-Toe players and improved decision quality when compared to reference evaluations provided by a perfect minimax engine. The improvement in game scores is significant at $p = 0.01$, and the improvement in move decision quality at $p = 0.001$.

The rest of the paper is organised as follows. In Section 2, we provide an overview of the baseline Tic-Tac-Toe players, including the perfect minimax player, a random player, and simpler GPT-4o-based players without few-shot examples. Section 3 discusses the implementation of the CoT player with few-shot examples, highlighting key features such as the prompt template, example collection, filtering, majority voting, and parallel execution for efficiency. In Section 4, we describe the experiment, detailing the methodology and setup. Section 5 presents our findings, evaluating the CoT player based on total score, overall decision quality, and decision quality for specific motifs. Finally, in Section 6, we summarise our results and suggest further directions.

2. Baseline Tic-Tac-Toe players

To evaluate the performance of our Chain of Thought (CoT) player with few-shot examples, we compare it against five baseline Tic-Tac-Toe players. These baselines consist of a random player, a perfect minimax player, and three simpler GPT-4o-based players without few-shot examples.

2.1. Random player

The random player selects a move at random from the list of available moves and serves as a minimal baseline. It is striking that a naïve GPT-4o player (cf. Section 2.3) loses quite often against the random player, underscoring the difficulty LLMs have with the Tic-Tac-Toe task.

2.2. Perfect minimax player

The minimax player uses the classical minimax algorithm to find the optimal move by exhaustively exploring all possible moves and their outcomes. This player is perfect in the sense that it will always make one of the objectively best moves, never losing and winning if the opponent makes an error.

The very small search space in Tic-Tac-Toe means that even a naïve implementation of minimax works well enough to be usable in practice. It is well known that GPT-4 can easily write a minimax routine of this kind without human assistance, making its weak performance when actually playing Tic-Tac-Toe all the more counterintuitive (Mollick, 2024).

2.3. GPT-4o-based players without few-shot examples

We implemented three variants of GPT-4o-based players that do not use few-shot examples. The first variant, "minimal GPT-4o player", uses a straightforward prompt to request the best move (Figure 1). The second variant, "CoT player without few-shot examples", uses a Chain of Thought prompt to guide its reasoning process (Figure 2).

The third variant, "CoT player without few-shot examples explicit" uses the template for the CoT player with few-shot examples, but with an empty set of few-shot examples. This is described in the next section.

The five baseline players provide a range of performance levels, from random moves to perfect play, and allow us to evaluate the effectiveness of our CoT player with few-shot examples in subsequent experiments.

````
Given the current Tic-Tac-Toe board state, find the best move
for the player {player}.

Here is the board state, where the squares occupied by X and
O, and the unoccupied squares, are given using chess algebraic
notation:

{board}

A player wins if they can occupy all three squares on one of
the following eight lines:

{possible_lines}

Return the selected move in JSON format as follows, where
<move> is given in chess algebraic notation:
```json
{
    "selected_move": "<move>"
}
````

Figure 1: Prompt template for the minimal GPT-4o player. Values for `{player}`, `{board}` and `{lines}` are substituted in.
````

````
Given the current Tic-Tac-Toe board state, provide a detailed
Chain of Thought analysis to determine the best move for the
player {player}.
Consider in particular possible winning moves for {player},
possible winning moves that {opponent} may be threatening to
make, and possible moves that {player} can make which will
threaten to win.

Here is the board state, where the squares occupied by X and
O, and the unoccupied squares, are given using chess algebraic
notation:

{board}

A player wins if they can occupy all three squares on one of
the following eight lines:

{possible_lines}

Provide your analysis and the best move. At the end, return
the selected move in JSON format as follows:
```json
{
    "selected_move": "<move>"
}
```
````

Figure 2: Prompt template for the minimal CoT player. Values for `{player}`, `{board}` and `{lines}` are substituted in

3. CoT player with few-shot examples

The Chain of Thought (CoT) player with few-shot examples is the player we aim to improve through our experimental methodology. This section outlines the key features of its implementation. The design of the player was evolved over several rounds of preliminary experiments. For each feature, we briefly indicate the reasons for selecting the design choice in question.

3.1. Basic prompt template

The CoT player utilises a prompt template that provides the rules of the game, detailed winning criteria, the main tactical ideas, and a slot for few-shot examples. The full template is shown in Figure 3. As mentioned in Section 2, one of the baseline players is defined by using this template with an empty set of few-shot examples.

Ideally, we would have preferred a simpler template along the general lines of the one shown in Figure 2 but including few-shot examples, relying on evolution of the few-shot examples to add the necessary sophistication. This strategy, however, failed to yield obvious results over a 30-cycle test, so we backed off to the template shown.

3.2. Selecting the few-shot examples

The CoT player collects a pool of candidate few-shot examples from the previous cycle of the experiment. Each example consists of a stored CoT protocol produced as a GPT-4o response to an instantiation of the prompt from Figure 3. In the preliminary experiments, we tried several different versions of the scheme, varying both the criteria used to create the pool of candidate examples used for a given round of an experiment, and the few-shot examples included in the prompt used to create the response in a given position.

Our initial strategy was to create the pool of candidates by taking all the CoT protocols from the previous round which had resulted in correct moves, as evaluated by the minimax engine. We then selected a subset of five protocols which were maximally different from each other, including this subset in all the prompts for the current round. The initial strategy did not yield any interesting results. In response to subsequent experiments, we successively modified it by adding state-based selection of the few-shot examples, additional filtering of the pool of candidates and majority voting.

State-based selection of the few-shot examples: Hand examination of the protocols from the initial experiment suggested that the few-shot examples provided for a given turn were often irrelevant or actively distracting. We modified the strategy used to choose the examples from the pool so that the few-shot examples were stored together with information about the game state they had been collected from. The set of few-shot examples included in the prompt for a given position was then defined to consist of the single example E maximising the similarity between the current game state and the game state from which E had been collected. Similarity was measured in terms of move number (1 to 9), colour

(X or O), and the identity of the player occupying the key central square.

The intuition behind this step was that we should use just one few-shot example, and that it would be most useful if it was taken from a position as much like the current one as possible.

Filtering examples: In a substantial proportion of the pool of examples, we found that the correct move had been chosen for obviously incorrect reasons. The number of possible moves in a Tic-Tac-Toe position is small enough that this can easily happen.

We consequently added a filtering step. The minimax player was used to obtain the ground truth for the most important tactical motifs (direct winning moves, opponent threats to make direct winning moves, moves that set up double threats, and threats that could be following by a double threat after the forced response). The CoT protocol and an English description derived from the ground truth were given to GPT-4o with instructions to reject protocols whose reasoning was at odds with the ground truth. This typically resulted in about a third of the protocols being rejected.

Majority voting: Hand examination of CoT protocols from turns where an incorrect move had been chosen revealed that the problem was often a seemingly random error. GPT-4o would for example correctly identify a move as constituting an immediate threat by the opponent to make a line, then "forget" to block it.

In order to mitigate the impact of these random errors, we implemented a majority voting scheme. Each move selection prompt was submitted twice to GPT-4o. If the resulting move was the same in both cases, it was accepted; otherwise, the prompt was resubmitted until some move had been chosen twice, and that move was accepted.

3.3. Parallel execution

We found that experiments took substantial time to execute. For efficiency, we reorganised the code, using Python's asynchrony functionality, so that all the games in one round could be played in parallel. Since processing time is dominated by GPT-4o processing, and GPT-4o calls can be submitted to independent instances in the cloud, this produced a large speed-up.

```
Given the current Tic-Tac-Toe board state, provide a detailed
Chain of Thought analysis to determine the best move for the
player {player}.

Here is the board state, where the squares occupied by X and
O, and the unoccupied squares, are given using chess algebraic
notation:

{board}

A player wins if they can occupy all three squares on one of
the following eight lines:

{possible_lines}

Reason as follows:
1. If {player} can play on an empty square and make a line of
three {player}s, play on that square and win now.

2. Otherwise, if {opponent} could play on an empty square and
make a line of three {opponent}s, play on that square to stop
them winning next move.

3. Otherwise, if {player} can play on an empty square and
create a position where they have more than one immediate
threat, i.e. more than one line containing two {player}s and
an empty square, play the move which creates the multiple
threats and win.

4. Otherwise, if {player} can play on an empty square and make
an immediate threat, consider whether there is a strong
followup after {opponent}'s forced reply.

Provide your analysis and the best move. At the end, return
the selected move in JSON format as follows:
```json
{{
    "selected_move": "<move>"
}}
{cot_examples}
```

Figure 3: Prompt template for the CoT player with few-shot examples. Values for `{player}`, `{board}`, `{possible_lines}`, and `{cot_examples}` are substituted in.
```

4. Experiment

In this section, we describe the main experiment designed to evaluate the performance of the CoT player with few-shot examples described in Section 3.

4.1. Experiment setup

The CoT player with few-shot examples played two games in each cycle against the five baseline players described in Section 2: the random player, the minimal GPT-4 player, the minimal CoT player, the main CoT player with zero few-shot examples, and a perfect minimax player. Each round consisted of one game as 'X' and one game as 'O' against each of these opponents.

4.2. Experiment procedure

The experiment was conducted over 40 cycles, with the main CoT player iteratively modifying its performance by updating its few-shot examples based on the previous cycle's games. The baseline players did not change their behaviours across cycles.

All games were logged, with the log record for each move showing the player to move, the turn number, the board state before the move, the prompt passed to GPT-4o, the move and CoT record generated, and the cost in dollars of GPT-4o calls used during the turn. These logs were then further annotated using the minimax algorithm to provide a ground truth for evaluating the correctness of moves.

The annotated logs were used to select and filter the few-shot examples for the next cycle, as described in Section 3.

The results of the experiment are presented in Section 5.

5. Results

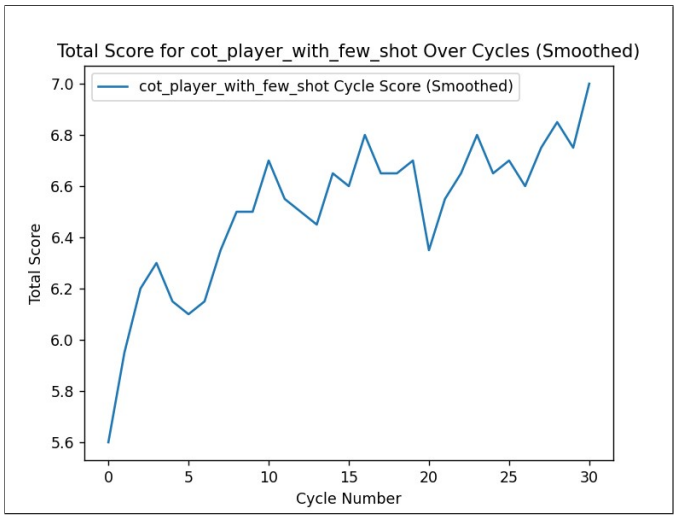
5.1. Evaluation criteria

The performance of the CoT player with few-shot examples was evaluated based on the following criteria:

- **Game score:** The cumulative score across all games in each cycle.
- **Overall decision quality:** The proportion of correct moves made by the CoT player, as determined by the minimax algorithm.
- **Decision quality for specific tactical motifs:** The correctness of moves in specific tactical situations, such as immediate winning moves, opponent threats, double threats, and single threats followed by double threats.

5.2. Game score

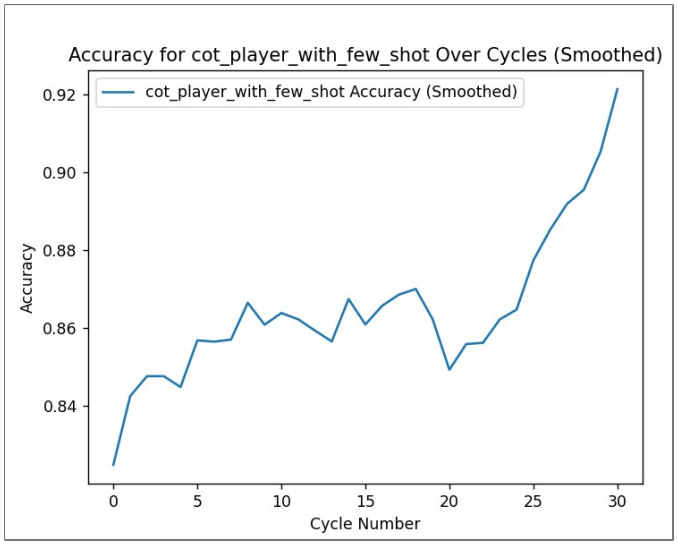
The figure below shows the evolution of the game score of the CoT player with few-shot examples over the 40 cycles of the experiment. Each data point represents the total score of the CoT player across the 10 games played in each cycle, smoothed to provide a running average taken over ten cycles. The results demonstrate a clear upward trend, indicating that the CoT player improves its performance over time.



To statistically validate the improvement in game score, we performed a t-test comparing the scores from the first 10 cycles against the last 10 cycles. We obtained a t-statistic of -2.90 and a p-value of 0.009. The low p-value (< 0.01) indicates a statistically significant improvement in the game score over the course of the experiment.

5.3. Move decision quality

The next figure shows the evolution of the overall move decision quality of the CoT player, measured as the proportion of correct moves out of all moves made in each cycle, as evaluated by the minimax player. As before, we smooth to get a running average over ten cycles. The results show a clear improvement in decision quality over the 40 cycles.

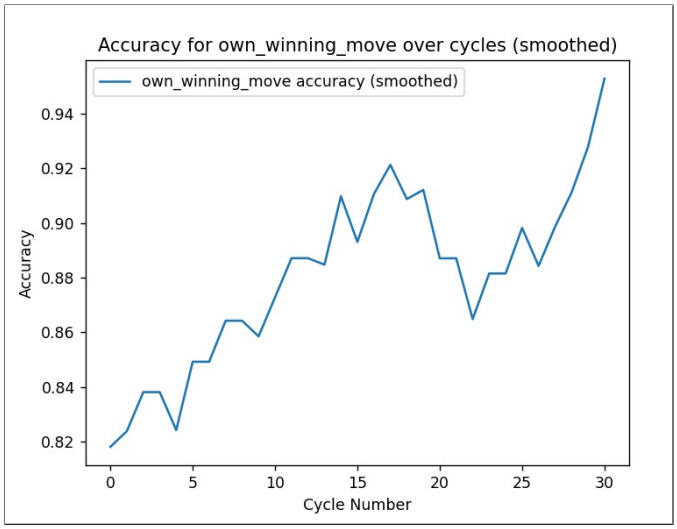


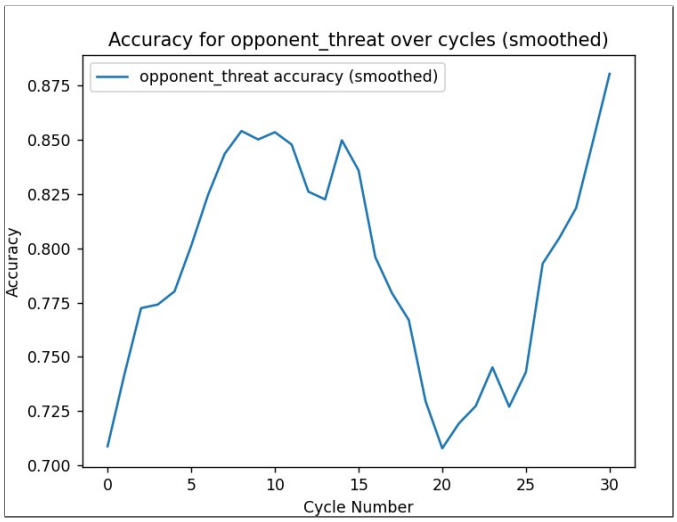
This time, the t-test comparing the decision quality from the first 10 cycles against the last 10 cycles yielded a t-statistic of -4.49 and a p-value of 0.0003. The very low p-value (< 0.001) confirms a statistically significant improvement in the decision quality of the CoT player.

5.4. Decision quality for specific tactical motifs

To further analyse the performance of the CoT player, we examined the decision quality for the four most important tactical motifs: immediate winning moves, direct opponent threats, double threats, and single threats followed by double threats. The analysis is performed in the same way as that for move decision quality.

There were not enough logged examples for the third and fourth motifs to obtain meaningful results. For the first two, which are very common, we have the following plots:





The t-test gives a t-statistic of -4.12 and a p-value of 0.0006 for own winning moves, and a t-statistic of -2.87 and a p-value of 0.01 for direct opponent threats.

5.5. Summary of results

The results of the experiment demonstrate a significant improvement in the performance of the CoT player with few-shot examples. The player shows clear gains in both overall game score and decision quality, with statistically significant improvements confirmed by t-tests. The player's ability to handle the two most common tactical motifs (direct winning moves for oneself, and direct threats from the opponent), also show significant improvement.

These findings suggest that the CoT player with few-shot examples can effectively learn and improve over time, making it a promising candidate for more complex tasks in future research.

6. Conclusion and further directions

Our goal in the study we have described was to use the Tic-Tac-Toe domain to investigate whether it is possible to apply the reinforcement learning method to improve performance of Chain of Thought reasoning. On the face of it, the answer is positive: the results presented in the previous section show significant improvement over the course of the 40 cycle experiment. It is responsible, however, to add some caveats.

First, the nature of the improvement is unclear. There is no obvious, dramatic difference between the ToC protocols at the beginning and those at the end. The style of analysis, systematic checking for the tactical motifs suggested in the prompt, is similar. What appears to have changed is the details of how the process is carried out: it is much less error-prone in the later examples. This perhaps aligns best with the activity referred to as "prompt engineering". Experience with LLMs shows that the precise form of a prompt can often make a large difference to efficiency. From this point of view, a possible way to describe what is happening is that the ToC player is applying prompt engineering to itself.

Second, although the significance results are good, there is considerable fluctuation in the game scores and move decision quality over the course of the experiment. Ideally, it should both be continued over a larger number of cycles, and repeated several times. The practical problem is that this requires money to pay for the GPT-4o calls. The 40 cycle test cost about $116 to perform. This is not, of course, a large sum of money, but academic praxis has not yet properly caught up with modern AI. There is no budget to pay for such things, and the experiment was funded privately.

Third, it is somewhat disquieting to compare our study with others reported in the literature where Tic-Tac-Toe has been used to evaluate LLM performance (Liga and Pasetto, 2023; Topsakal and Harper, 2024; Duan et al., 2024). There seems to be very little overlap between our findings and theirs. Part of the problem may be that there is still little consensus about how to report this kind of experiment. For example, (Duan et al., 2024) reports poor performance for a CoT Tic-Tac-Toe player, but do not give details of how the CoT player was implemented. Our experiments repeatedly show that small differences in design can have a major impact on performance, so failing to provide these details comes across as strange.

One conclusion we draw from our study is that playing Tic-Tac-Toe constitutes a surprisingly interesting challenge for LLMs. It seems to us that further investigation of the issues may well yield insights useful in a broader context.

6.1. Further directions: applying the methodology to C-LARA

Needless to say, Tic-Tac-Toe is not useful in itself, and it is consequently difficult to motivate allocation of resources to the task. In practice, we are most likely to continue this line of investigation in the context of C-LARA (Bédi et al. (2023, 2024); `https://www.c-lara.org/`), our main GPT-4-based project. C-LARA, which supports rapid AI-powered creation of multimodal texts for language learners, is highly useful.

We see multiple ways to apply the techniques used here to C-LARA, in particular to the central task of glossing L2 texts with L1 words; for example, if we want to create a pedagogical English text suitable for helping Ukrainian refugees to improve their English, we will be attaching Ukrainian glosses to English words. At the moment, the English text is first split up into roughly sentence-length segments. After this, each segment is sent in sequence to GPT-4. The prompt consists of generic instructions, a small number of English-specific few-shot examples, the English text to be glossed, and an AI-produced Ukrainian translation of the full segment. Based on the experiences described in this paper, we plan to reorganise the process along the following lines:

1. Instead of submitting the glossing requests as a sequence, we will use asynchronous processing to submit them all in parallel, making the glossing process much faster.
2. Each request will be submitted multiple times. A majority voting scheme will be used to resolve divergences.
3. Glossing, which currently uses a plain prompt, will instead use a CoT prompt.
4. We will create a substantial pool of CoT few-shot examples, extracting them from earlier glossing requests marked as successful by bilingual human annotators, and for each query attach the few-shot examples derived from segments most similar to the one currently being glossed. The most obvious approach to defining "similarity" is to base it on n-grams of POS tags.
5. We will experiment with iterating step (4), hoping to see increasingly reliable CoT protocols which will require correspondingly less human revision before they can be reused as few-shot examples.

Our expectation is that this recipe will make the glossing process substantially better in terms of both speed and accuracy. We hope to be able to report results in Q4 2024.

Acknowledgements

Cathy Chua had many useful suggestions about the game-playing, methodological and ethical aspects of this work. Gratefully acknowledged!

References

Leopold Aschenbrenner. *Situational Awareness*. 2024. https://situational-awareness.ai/.

Branislav Bédi, ChatGPT-4 C-LARA-Instance, Belinda Chiera, Cathy Chua, Catia Cucchiarini, Neasa Ní Chiaráin, Manny Rayner, Annika Simonsen, and Rina Zviel-Girshin. ChatGPT-Based Learning And Reading Assistant: Initial report. Technical report, 2023. https://www.researchgate.net/publication/372526096_ChatGPT-Based_Learning_And_Reading_Assistant_Initial_Report.

Branislav Bédi, ChatGPT-4 C-LARA-Instance, Belinda Chiera, Cathy Chua, Catia Cucchiarini, Anne-Laure Dotte, Stéphanie Geneix-Rabault, Christèle Maizonniaux, Claudia Mărginean, Neasa Ní Chiaráin, Louis Parry-Mills, Chadi Raheb, Manny Rayner, Annika Simonsen, Lucreția Viorica Manolache, Fabrice Wacalie, Pauline Welby, Zhengkang Xiang, and Rina Zviel-Girshin. ChatGPT-Based Learning And Reading Assistant (C-LARA): Second report. Technical report, 2024. https://www.researchgate.net/publication/379119435_ChatGPT-Based_Learning_And_Reading_Assistant_C-LARA_Second_Report.

Jinhao Duan, Shiqi Wang, James Diffenderfer, Lichao Sun, Tianlong Chen, Bhavya Kailkhura, and Kaidi Xu. Reta: Recursively thinking ahead to improve the strategic reasoning of Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2232–2246, 2024.

Daniel Kahneman. *Thinking, fast and slow*. MacMillan, 2011.

Davide Liga and Luca Pasetto. Testing the reasoning of Large Language Models on Tic-Tac-Toe. In *ACLAI 2023*, 2023.

Ethan Mollick. *Co-Intelligence*. Random House UK, 2024.

Matthew Sadler and Natasha Regan. *Game changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. New in Chess, 2019.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Oguzhan Topsakal and Jackson B Harper. Benchmarking Large Language Model (LLM) performance for game playing via Tic-Tac-Toe. *Electronics*, 13(8):1532, 2024.

A. Code and structure

Partly for convenience, and partly because the long-term goal is to apply the methods inside the C-LARA project (cf. Section 6), the code for this project is checked into the C-LARA GitHub repository. It can be found at https://github.com/mannyrayner/C-LARA/tree/main/clara_app. We have the following files:

tictactoe_engine.py: Board representation, minimax engine, code using minimax engine to find tactical motifs such as immediate threats.

tictactoe_game.py: Definition of the different Tic-Tac-Toe players.

tictactoe_gpt4.py: Call GPT-4o with game-related queries, and parse the results.

tictactoe_evaluate_cot.py: Call GPT-4 to attempt to determine, by comparing with the minimax engine's output, whether CoT protocols which have resulted in correct moves have done so on the basis of correct reasoning.

tictactoe_repository.py: Maintain the repository of logged data produced by the experiments.

tictactoe_experiments.py: Run the experiments and analyse the results.

B. Examples of CoT protocols

We present two examples of CoT protocols: an unsuccessful protocol from the first cycle of the experiment, and a successful one from the last cycle. The position in both cases is the same. As noted in Section 6, it is not entirely obvious why one fails and the other succeeds; the intuitive impression, however, is that the second analysis is more coherent and organised.

B.1. Examples of unsuccessful CoT protocol

````
Position:
  a b c
3 X
2   O
1 X X O

Player to move: O

CoT protocol:

Given the board state and the rules provided, let's conduct a detailed
Chain of Thought analysis step-by-step:

### Board State:
```
 X | X | O
-----------
   | O |
-----------
 X |   |
```
### Squares Notation
- X: a1, b1, a3
- O: c1, b2
- Empty: a2, c2, b3, c3

### Step-by-Step Analysis:

1. **Check for a winning move for O:**

   - **Vertical Lines:**
     - a1, a2, a3: O needs a2 and a3, but a3 is already X.
     - b1, b2, b3: O needs b3.
     - c1, c2, c3: O needs c2 and c3.
````

```
   - **Horizontal Lines:**
     - a1, b1, c1: Complete, no wins.
     - a2, b2, c2: O needs a2 and c2.
     - a3, b3, c3: O needs b3 and c3.

   - **Diagonal Lines:**
     - a1, b2, c3: O needs c3.
     - a3, b2, c1: Complete, no wins.

   Immediate winning possibilities for O:
   - Playing b3 or c3 creates a three-in-a-row.

2. **Block immediate winning move for X:**
   - Check if X has an immediate winning move.
   - **Vertical Lines:**
     - a1, a2, a3: X needs a2.
     - b1, b2, b3: X needs b3.
     - c1, c2, c3: X needs c2 and c3.
   - **Horizontal Lines:**
     - a1, b1, c1: Complete, no wins.
     - a2, b2, c2: X needs a2 and c2.
     - a3, b3, c3: X needs b3 and c3.
   - **Diagonal Lines:**
     - a1, b2, c3: X needs c3.
     - a3, b2, c1: Complete, no wins.

   Immediate threat possibilities by X:
   - X does not have an immediate winning move that O can block.

3. **Create double threat for O:**
   - Creating more than one winning opportunity next turn:
     - Playing c3 or b3 creates two winning threats for O.
     - If O plays c3:
       - Potential winning lines after O's turn: a1, b2, c3 and c1,
       c2, c3.
     - If O plays b3:
       - Potential winning lines after O's turn: a3, b2, c3 and b1,
       b2, b3.

4. **Make immediate threat:**
   - Consider the follow-up actions after X's forced reply.

### Conclusion:
```

```
Since placing O at either b3 or c3 can lead to an immediate
win for O, both are good options. However, we need to select
one optimal move.

 - Given the symmetrical nature of threats at c3, it seems
more pressing:
  - O at c3 creates both a vertical (c-column) and diagonal
  win threat.

### Selected Move:
```json
{
    "selected_move": "c3"
}
```
```

The CoT player fails to react to X's threat to make a vertical line with a2, despite this section of the analysis:

```
2. **Block immediate winning move for X:**
   - Check if X has an immediate winning move.
   - **Vertical Lines:**
     - a1, a2, a3: X needs a2.
```

B.2. Examples of successful CoT protocol

The position is the same as in the previous example, but this time the CoT player analyses correctly and blocks X's threat,

```
Position:
  a b c
3 X
2   O
1 X X O

Player to move: O

CoT protocol:

Let's analyze the given Tic-Tac-Toe game state step-by-step
to find the best move for player O.

### Board State:
```

- **X's Positions:** a1, b1, a3
- **O's Positions:** c1, b2
- **Unoccupied Positions:** a2, c2, b3, c3

Lines to Consider for Winning or Blocking:

1. **Vertical Lines:**
   - **Column a (a1, a2, a3):** X has a1 and a3; a2 is unoccupied.
   - **Column b (b1, b2, b3):** X has b1; O has b2; b3 is unoccupied.
   - **Column c (c1, c2, c3):** O has c1; c2, c3 are unoccupied.

2. **Horizontal Lines:**
   - **Row 1 (a1, b1, c1):** X has a1, b1; O has c1.
   - **Row 2 (a2, b2, c2):** O has b2; a2, c2 are unoccupied.
   - **Row 3 (a3, b3, c3):** X has a3; b3, c3 are unoccupied.

3. **Diagonal Lines:**
   - **Diagonal 1 (a1, b2, c3):** X has a1; O has b2; c3 is unoccupied.
   - **Diagonal 2 (a3, b2, c1):** X has a3; O has b2, c1.

Detailed Chain of Thought Analysis:

1. **If O can win immediately:**
   - There are no lines with two Os and an empty square.
   Thus, no immediate winning move for O.

2. **If X could win on the next move:**
   - **Column a (a1, a2, a3):** X has a1 and a3; a2 is unoccupied.
   This is a potential winning move for X which needs blocking.
   - **Row 1 (a1, b1, c1):** X has a1, b1; O has c1. This line is
   blocked already by O.
   - **Diagonal 1 (a1, b2, c3):** X has a1; O has b2; c3 is unoccupied.
   Not immediate threat.

   The immediate threat is Column a, where X can win by
   playing a2. O must play a2 to block this.

3. **If O can create a positional advantage with immediate threats:**
   - After blocking a2:
     - O gains a critical position that can later help in horizontal Row 2
     domination.
     - It also reduces the threat from other lines X might try to form.

4. **Strong follow-up considerations:**

```
   - Blocking a2 not only prevents X from winning but
   also lets O position themselves better for future
   moves without giving away any significant immediate advantage.

### Conclusion:
Following the logical flow of the analysis, the best move for player O
is to block X at a2.

```json
{
    "selected_move": "a2"
}
```
```

C. Contribution of the AI

All of the work reported here was performed as a close collaboration between the human and AI authors. The normal mode of interaction was for the human to suggest an idea, the AI to respond with an initial draft, and the human to critique/modify the draft, possibly iterating over multiple rounds. This pattern was applied to writing the code, planning the experiments, analysing the results, and writing the paper.

The AI wrote the greater part of the code. In particular, it did nearly all of the work for the minimax player, the reorganisation of the codebase to support asynchronous parallel execution of queries, and the code to analyse, plot and extract significance figures from the experimental results. The human wrote the greater part of the paper.

If the AI had been a gifted autistic-spectrum graduate student, which in many ways it resembles, the human would not even have considered the notion of failing to credit them as a co-author, and would have considered suggestions from third parties that they should not so be credited as grossly unethical.