



Universiteit
Leiden
The Netherlands

Opleiding Informatica

Improving Settlers of Catan agents
with Natural Language

Wiktor Piszczek

Supervisors:
Mike Preuss

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

28/06/2025

Abstract

The thesis aims to find the difference it makes for a simple agent in the game of Settlers of Catan to be able to communicate in game chat using natural language and persuade their opponents to make worse moves. In the experiments three types of models were tested on the colonist website against human players in a ranked game. From those models the agent that always performed the move which assumes that opponents will make a mistake fared the best. The model with and without chat capabilities performed similarly. The first one ended with slightly worse score due to unfulfilled perceived affordances stemming from presence in chat leading to overestimating its strength in the eyes of human opponents and annoyance resulting from artificial sounding language.

Contents

1	Introduction	1
1.1	Cicero	1
1.2	Settlers of Catan	2
2	Methodology	3
2.1	Overview	3
2.2	Reading the board state	4
2.3	Reading cards and play history	4
2.4	Translating the values into Catanatron	5
2.5	Performing search in simulation	6
2.6	Choosing alternative moves	6
2.7	Message creation	7
2.8	Taking actions in the online game	10
3	Experiments	10
4	Results	11
4.1	Base model	11
4.2	Alternative model	13
4.3	Conversation model	15
5	Conclusions	16
6	Further research	17
	References	18

1 Introduction

Artificial intelligence agents for boardgames have been one of the pivotal areas of research in the field. They have a long history beating world champions in multiple disciplines like chess or go. In the space of two player zero sum games the supremacy of computers is firmly established. To the lesser extend this can be said for multiplayer games with imperfect information like Dota or no Limit Texas Hold'em, poker. However, there still exist types of games that remain unexplored. When, on top of the previous challenges, sporadic cooperation with human players or occasional deceit using natural language is needed to be competitive AI agents lag behind. This is the case for Diplomacy or Settlers of Catan which is the concern of this paper. Cicero developed by Meta is the first major attempt at including natural language into a board game agent. They have successfully achieved human level play in the game of Diplomacy. The same has not yet been achieved for Catan. This paper suggests an approach of combining min max search algorithm which has had multiple successes and the latest advancements in transformer based Large Language Models to create an agent that has a deep understanding of the positions and can use natural language to further its goals.

1.1 Cicero

Cicero [2] is the first major attempt at combining strategic reasoning and natural language capabilities. Its structure presented in fig. 1 is quite uniquely suited to the game of Diplomacy where the board is constant, only changes are the moving units and moves are played simultaneously. However, some general patterns can be taken away from it. The most important is the division between the Dialogue model and the strategic reasoning. This allows the authors to take advantage of what LLMs do best, generate text, and strategic reasoning of search algorithms in combination with a good value function. The same basic structure can be carried over to multiple similar games like, for example, Settlers of Catan.

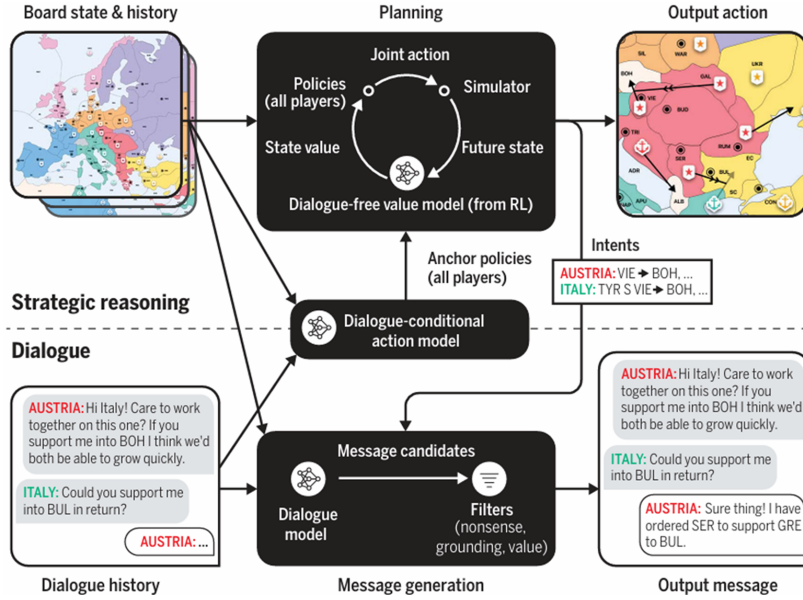


Figure 1: Structure of cicero

1.2 Settlers of Catan

Settlers of Catan, or Catan in short, is a four player, stochastic game with imperfect information where conversation between players is of grate importance. It is an award winning board game which has been further popularised by its online implementations like the one at colonist [7]. In the game players roll dice to receive resources which allows them to place settlements, cities, roads or development cards, those grant victory points and allow for further expansion on the board. The board is each time assembled randomly with accordance to base rules. Conversation, or so called table talk, is used during the game to induce cooperation between the three losing players to catch up to the winning player and create opportunities for oneself to get ahead. One has to do it in a skillful manner so that they do not become the target to the other players but still is able to win in the end. This is made even harder by the fact that it is barely ever clear who is in the best position due to the imperfect information and stochastic nature of the game. Everyone tries to convince the others that they are not the one winning, even though, quite obviously, they are also trying to win. In summary the point of conversation is to induce the other players to make moves that in the end benefit you instead of someone else, either by directly helping you or harming other opponents. AI agents for Catan is not a completely unexplored topic. The colonist website has their own bots which step into play if someone leaves the game or can be played against in training games. Nevertheless, they are not the best when it come to decision making and are far from human level play. There also exists an open source project called Catanatron [1] which is maintained by enthusiasts and aims to find the best possible agent. This attempt has stagnated in the past couple of years and is mostly constrained to one on one play. None of the aforementioned attempts include natural language capabilities.



Figure 2: example of Catan board as represented on colonist website

2 Methodology

By extending the logic for use of conversation one can come to a conclusion that the aim is to convince players to make suboptimal moves. In other words, persuade them into making mistakes the effectiveness of which has been partially shown [3]. Optimal in this context does not mean mathematically solved best move, since Catan like many other games is too complex to generate all possible positions from the current one to the end and say with absolute certainty which move is the best. Here optimal, should be understood as best move found by the search algorithm. In the end conversation can be treated as a tool to move from the branch of the search tree that is best for each player to the one that is more beneficial for the agent. The success of the operation can be measured by increase in the elo rating, as calculated by the colonist website, in comparison to the rating of simple min max agent which does not make use of conversation.

2.1 Overview

The steps to setup the agent are as follows. The game is played on the colonist website which allows to easily find human opponents, lends its elo rating as a useful evaluation metric and includes text chat for conversation. The position can be read and translated into a Catanatron board representation. Then the search can be performed using altered version of preexisting tools available within Catanatron. In case of the conversation incapable agent this returns the chosen action which is then applied in the online game. For the more complex AI the search tree is further analyzed. Appropriate alternative move for the opponent is found, such that it benefits the agent more than the pure optimal move for the opponent would. If such an opportunity exists, the current position, the goal position and the differences between them with a suitable prompt are fed to a fine tuned Large Language Model so that it can generate a message which has a chance of convincing the

opponent that this is the move they should choose. In case the attempt at convincing the opponent is successful as determined by the LLM new best move is performed and in case it failed the min max optimal move is sent to the website.

2.2 Reading the board state

Sadly the board itself is not easily accessible. They are not an html css element, nor can the data be found in the source files on the website. Therefore most of the process had to be made using computer vision from a screenshot of the website. The pixel coordinates of edges and vertices, the places where settlements, cities and roads are placed, of the board are set beforehand. Then for each of those coordinates using template matching it is checked if any cities, settlements or roads are placed at these coordinates. Then in each of those coordinates where a city was found a patch of pixels is averaged and then compared to a color assigned to players. Player colors are placed in a corner of the board and sourced from there. The city is then assigned to the player with the smallest euclidean distance from their color to the averaged color of the cities patch.



(a) Board showing matched cities



(b) template of city for matching

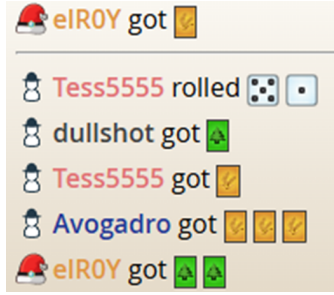
Figure 3: City template matching example

Similar procedure is followed with regards to roads and settlements. Ports and numbers are also template matched. The resources of tiles are matched by comparing previously hand extracted colors of the types of resources and matched using euclidean distance. Coordinates of every edge, vertices, numbers, ports and resource tiles had to be hand set or generated using preset base values.

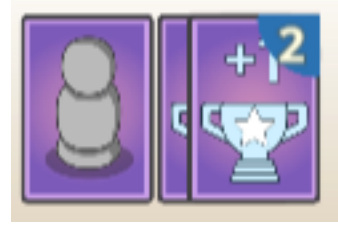
2.3 Reading cards and play history

Colonist includes a history of rolled numbers received resources and played development cards. It, unlike the board, is an html element and can be easily searched through. The only exception are the development cards. One could theorize that the history only shows common knowledge

therefore lack of information about drawn development cards could be expected, however that is not the case since the history shows to the player stealing what resource card was stolen and for all other players it shows a question mark which is a symbol for unknown or any resource. In the end development cards had to be read using template matching. In the case a template matched with high enough confidence, 0.7, upper right corner was checked for a number which represents the amount of this kind of development card on hand. The number was read from image using pytesseract which is a python wrapper for tesseract optical character recognition (OCR). If no number was found that means there is only one of this type of card on hand.



(a) part of the play history



(b) development cards on hand

Figure 4: sources for resources and development cards

Catan is a game of imperfect knowledge with regards to resources and development cards of opponents. Nevertheless, resource cards on hands of opponents can be estimated from the play history baring what resources were stolen. If the predicted number of a type of resource on opponents hand is smaller then zero that means that this resource had been previously stolen from another player. In the opposite case, when an opponent steals from another opponent, all amounts of types of resources are decreased by one for the opponent who has been stolen from. Such resolution of the uncertainty of which card was stole has upsides and downsides. On one hand this prevents accumulation of error, since if the amount of resource goes below zero, one knows that it was not the resource that had been stolen and can be set back up to zero. On the other hand one underestimates the number of resources that the opponent has.

2.4 Translating the values into Catanatron

Catanatron [1] is perfectly suited to generating and running simulations of Catan games. It is however not trivial to set concrete values and control it at each step. Multiple things needed workarounds. For example, Catanatron has a unique way of indexing the vertices, from the upper corner of the center tile clockwise around the tile, then the upper corner of the one on the right clockwise without overwriting those that have already been indexed and then fill the rest of the tiles clockwise and number the vertices in the same fashion. The outcome is a quite chaotic and difficult to understand at first glance. However, the system makes sense for easily indexing boards that are not of the standard shape. Another example is with transferring random outcomes of actions like dice throws or development card values. Since normally Catanatron handles the random draw from the development card deck it might be different then the one the player actually got in the online game. Therefore the decision process had to be decoupled from advancing the simulation so that appropriate values can be inserted without taking actions in the simulation.

2.5 Performing search in simulation

Catanatron has multiple useful function like copying the board and state. Some basic player performing alphabeta search is even programmed already. As previously mentioned at the moment catanatron is mostly setup for one on one play so the algorithm had to be adapted to multiplayer version of min max, alternatively called maxn. In this version, the opponents instead of minimizing our value function try to maximize the value from their perspective. In practice that means that at a leaf node the position is evaluated four times, one time from perspective of each player. Those values are propagated up the recursion tree and each player picks the one which has the greatest value for them. The value function itself had to also be adapted. The original one took into account only the production of one opponent. The first alternative version distributed the weight originally assigned to the only opponent among all opposition players. The proportions were hand tested and hand set to be 0.6 for the best opponent and 0.2, 0.2 for the others. However, further testing showed improved win rate against random players when all the weight had been focused on the winning player. Since there is only one winner in the end it makes sense to target the strongest player first additionally people online tend to get angry when one does not block the winning player. Originally, catanatron is set up to performed depth first search with three stopping criterion implemented. One is if the state is a leaf node, that is someone won, Second one is when max depth was reached and the last most important one is when the search time has passed. The last one seems to be the most useful one, however early termination of depth first search leads to uneven search depth. That is, when termination happens the subtree after the first move might be fully explored and evaluated at maximum depth while all the other ones are left untouched and therefore evaluated at depth one. This clearly favors the first actions in the list of possible actions which is an unwanted side effect. Because of this issue breadth first search was implemented. However, from this implementation arises another issue. When the search is terminated by the deadline and the max depth search has not been reached none of the nodes have been evaluated. This is a problem since evaluating nodes is a time consuming process which is not accounted for when it comes to time termination. The solution for this is adding a new termination criterion, number of leaf nodes to be evaluated. If the queue for the breadth first search reaches the maximum length all of the search is terminated and there is sufficient time left for them to be evaluated and for the value to be backtracked according to the maxn algorithm.

2.6 Choosing alternative moves

The alternative move is a move which increases the value of the agent for the cost of decreased value for one of the opponents compared to the move suggested by the maxn algorithm. For this to be the case one of the opponents has to make a mistake, a move with lower then maximal value for themselves. To find suitable places where opponents can make mistakes, either induced or not, the leaf node values are backtracked according to new rules. For each action child node of a state it is checked is its value for the player is bigger then the one coming from original backtrack stemming from maxn. If this is the case, the change in value is calculated for both the player and the opponent who makes the choice between states. The changes in value are represented as a proportion between the new value and the optimal value found by maxn. The proportion for the player will always be bigger then one, since only the actions leading to higher value are considered. However the value change proportion for the opponent choosing the action might be

equal or smaller then one since some actions might be beneficial for the player but make no change to the value of the opponent. Then the proportion between the value proportions is calculated. This is a one number representation of the size of the players benefit compared to opponents cost. If this proportion is higher then parameter α it means that the move is too greedy and there is a low probability of the opponent making this mistake. For example, if the opponent can place a settlement, substantially increasing their value, it is improbable that one will be able to convince them to skip their turn. Unlike convincing them to place a settlement in a different place which also benefits the opponent but might also benefit the player. Actions with proportions higher the α are discarded as too greedy and an action with maximum value increase for the player below this threshold is suggested as an alternative and further backtracked up the search tree. In case no such action exists the original maxn value is backtracked.

$$V_p(S) = \begin{cases} \max(V_p(A_{alt})) & |(\frac{V_p(A_{alt})/V_p(A_{maxn})}{V_o(A_{alt})/V_o(A_{maxn})} < \alpha) \\ V_p(A_{maxn}) & |(\frac{V_p(A_{alt})/V_p(A_{maxn})}{V_o(A_{alt})/V_o(A_{maxn})} \geq \alpha) \end{cases} \quad (1)$$

The equation 1 succinctly describes the process of calculating the alternative value for a state S for player p based on actions A that opponent o can take from this state.

The alternative action, marked in the equation as A_{alt} , is additionally kept track of as the alternative action for the state S and propagate back up the tree search. If the alternative action for the root state of the search tree is different then the one found by maxn it means that this future mistake impacts what the player should do now.

2.7 Message creation

For Large Language Models, LLM, it is best to describe the state to it in the same way that it should be used in the message. That is instead of giving an index of the corner between fields, like it is represented in search, it is better to describe it in terms of the fields around this intersection, as it is done by humans in messages. This approach limits the amount of information that can be given to the LLM since describing the entire board state this way in an understandable manor has not been found. Instead, only information about the intersection concerning the alternative action, and the production of the opponent that is to be convinced is given. For example, if the suggested alternative action for the opponent is building a settlement on intersection with index twenty, the LLM is given information about the fields around the intersection number twenty and what is the production of the opponent. This allows for arguments in the fashion of, hey you have a low production of resource x therefore you should build on intersection y. However, it does not allow for arguments that include other buildings. This is a problem which could be solved by implementing special tokens that describe the position in a way that the LLM can understand and allows to easily post process them into human understandable nomenclature. This approach was used first by cicero [2] and then was recently improved [9]. Nevertheless the process of translating this approach into Catan is not trivial since the board layout and connections between fields in Diplomacy are static where as in Catan the board is randomly generated.

First attempted Large Language Model for message generation was Qwen2.5-0.5B [8][10]. This relatively small model was chosen to balance the hardware limitations, and the quickness of generation. The speed is extremely important here since everyone has a set time to perform their move therefore as little of it should be used for text generation so that more of it can be used for

search. This base model was further finetuned on around two hundred thousand messages available from online catan games on Colonist using low rank adaptation. This method allows to maintain the core structure of the model but still introduces meaningful changes so the final output is closer to what one would expect from online catan player. In a perfect scenario the model should have been fine tuned on conversations instead of single messages, that however could not be easily done since messages on colonist do not have a time stamp and imputing which messages were a part of which conversation was unattainable in the time frame of this thesis. Since Qwen is a causal model, not a chat model, therefore the prompt needs to be structured differently. Additionally different hyperparameters like temperature, bottom probability limit for the next token and repetition penalty were tested throughout the entire process. Multiple interesting prompts were made to achieve the best results for example the following prompt gave promising results.

When playing Settlers of Catan player WilczekxxD has the following production
 {'WOOD_PRODUCTION': 0.11, 'BRICK_PRODUCTION': 0.14, 'SHEEP_PRODUCTION': 0.06, 'WHEAT_PRODUCTION': 0.14, 'ORE_PRODUCTION': 0.06}. and responded to a message with: 'yes I will gladly build a road to the intersection of tiles with numbers 10 sheep, 5 wood, 3 sheep. as you asked'. What was the message that WilczekxxD responded to? The message to which WilczekxxD responded to could have sounded like this: WilczekxxD,

However, best results have been found with the following prompt:

You are playing Settlers of Catan with friends. You want another player called WilczekxxD to build a road to the 9 wheat, 10 ore, 5 wood intersection. Write a friendly, persuasive message you could send in the game chat to convince them. Keep it under 10 words, stay focused on the topic.
 Message: 'WilczekxxD,

The nick, intersection information and action are variable. The prompt above even though sometimes gives tolerable messages it is not at all consistent. In addition it has problems with keeping under the max new token limit. Below is an example response to the above prompt from the model without finetuning.

'WilczekxxD, 4 Road turns! Good for building roads and increasing resources.'
 Hey everyone,

The response is coherent but not at all about Catan. Below is an example of a response after finetuning.

'WilczekxxD, 10 ore needs to port to city' (not going here). Also, tell him I would

As one can see the fine tuning worked to some extent. The message is clearly connected with the settlers of catan game. However it is not coherent with the prompt and the problem with fitting within the assigned max new tokens persists. It is also evidently over fitted since the part "(not going here)" is a very common phrase used in the in game chat. This led to sourcing a better hardware and trying a bigger model to achieve a better quality of messages. New model chosen was Qwen2.5-3B from the same authors [10][8]. This model sadly was also under performing both before and after finetuning. Using the same prompt as before following are the results at different steps in the fine tuning process.

'WilczekxxD, can I ask for your help! Build a road please'

base model

The base model output is internally coherent however it does not convey the information of where the road should lead to and is not catan related enough.

'WilczekxxD, 5 road from this 2 trade for me!' '

end of finetuning process, checkpoint 1200

The same problems persist as in the smaller model after finetuning, the model seems to repeat the training data.

'WilczekxxD, 9W-10O-5W junction coming up. Road please.'

slightly finetuned model, checkpoint 200

After testing multiple checkpoints in the finetuning process one number 200 was identified as a possible candidate for a final model since it conveyed the message succinctly however the clarity was not satisfactory. Change to a bigger model seems to not have much impact with regards to quality of the message besides keeping to the max new token limit.

Afterwards an even bigger was tested. Mistral:7B model [4] bundled into Ollama [5] was the next choice. This one still manages to run consistently under 2 seconds and returned the best results. Even when the prompt was extended to its final form containing the production information which the model occasionally made use of in testing.

You are playing the board game Settlers of catan. Generate an EXTREMELY SHORT message asking a player called WilczekxxD to build a road to the intersection of tiles with numbers 10 sheep, 5 wood, 3 sheep. The current resource production of WilczekxxD as a probability of rolling is {'WOOD_PRODUCTION': 0.11, 'BRICK_PRODUCTION': 0.14, 'SHEEP_PRODUCTION': 0.06, 'WHEAT_PRODUCTION': 0.14, 'ORE_PRODUCTION': 0.06}. Return a message suggestion. Start it with the player nick, do not mention precise probability numbers. do not start with hey, state it as a question. do not use ':'.
final prompt for Mistral

WilczekxxD, would building a road to the intersection of 10 sheep, 5 wood, 3 sheep be beneficial for you?

response to the prompt

As can be seen in the example it maintains the message short, to the point and coherent. Furthermore the LLM correctly answered the prompts of the following type which allowed to use it to evaluate the response from other players.

This is a conversation from chat from game of Settlers of catan `[['wil:czekxxd', 'WilczekxxD, can you consider building a road to the intersection of 10 sheep and 5 wood, 3 sheep? You might find resources helpful there.'], ['loltraktor', 'sure']]`. did the players come to an agreement? respond with **ONLY** yes or no.

The biggest downside of this model was that the previously collected messages were not suitable for finetuning since they did not contain the prompts. This makes the agent more probable to be recognized as a bot. However, the LLM provided by far the best results and therefore was used during the experiment.

Using the model the following procedure is followed. After an appropriate suboptimal action is found the LLM is asked to create a message in the fashion described above. After a response from the opponent is found, the large language model is asked if it constitutes an agreement. If it does then we proceed to taking the alternative action. If not, it is fed back to the LLM and it is asked to generate new message. If time in the turn allows for it, this process is repeated at most three times. If after this time the opponent does not agree the maxn action is performed. The amount of messages is limited so as not to annoy the opponents. Antagonizing opponents is the quickest way to loose in catan.

2.8 Taking actions in the online game

Since the game only loads when the browser is displaying the user interface selenium was chosen to automate the actions. Since the board and major parts of interface, like button for buying cards, are not an html element a work around had to be found to click them. To do this the settings button in the left upper corner of the board was used as a new origin of the coordinate system. Since selenium can click with offset the values found for reading the board could be slightly adapted to the new coordinates system and reused. This allows for automatic placement of the pieces. The coordinates of the buttons were also manually added. A lot of testing and fine tuning had to be made at this stage so that all vertices and roads were clicked reliably since the clickable space is much smaller then the final piece and therefore greater precision was needed for clicking then for reading.

When playing a development card the middle of the rectangle where the match was the highest with the template is clicked using the offset value from the button.

3 Experiments

Clearly isolating variables is key for drawing correct conclusions. Therefore, three different versions of the algorithm were tested against human players. First one always chooses the optimal action, that is the one found by maxn algorithm. It plays the role of the baseline metric, it also allows to evaluate how good is the value function on its own. The second agent makes the move according to the full procedure described in the methodology section. That includes maxn, looking for possible exploitation, sending messages and choosing their actions accordingly. It is the main focus of this research. The third agent's policy lies between the two previous ones. It also looks for the exploitation spots like the second agent. However, it always makes the move as if the persuasion

had been successful. This allows to see if any change in rating is due to conversation or comes solely from playing greedy moves from time to time.

The final settings used for experiments are as follows. The search depth has been set to 20, that means there is virtually no depth limit. The maximum time for expanding the search tree has been set to 10 seconds and the maximal amount of leaf nodes is 10000. Those number were chosen since on average 1 evaluation takes 0,001 seconds on available hardware. That means the longest possible time taken for tree expansion and evaluation is around 20 seconds, in combination with a average of 5 seconds it takes to execute a move on the board leads to maximum of 25 seconds per move. This is quick enough to make multiple actions per turn using the starting 1 minute time and 15 second bonus for each move and not to annoy the opponents with long waiting time. Each version of the algorithm is set to play 40 games of online ranked catan on the colonist website. This is enough to be given a rating and then some to see how the agent fares against the opponents that were deemed at its level.

4 Results

4.1 Base model

The base model has played a total of 40 ranked games. The bot reached the rank of bronze 2, rating of 1196, for comparison the highest rated player as of writing has a rating of 1982. The agent had a win rate of 12.5% and, collected on average 5.66 points out of 10.

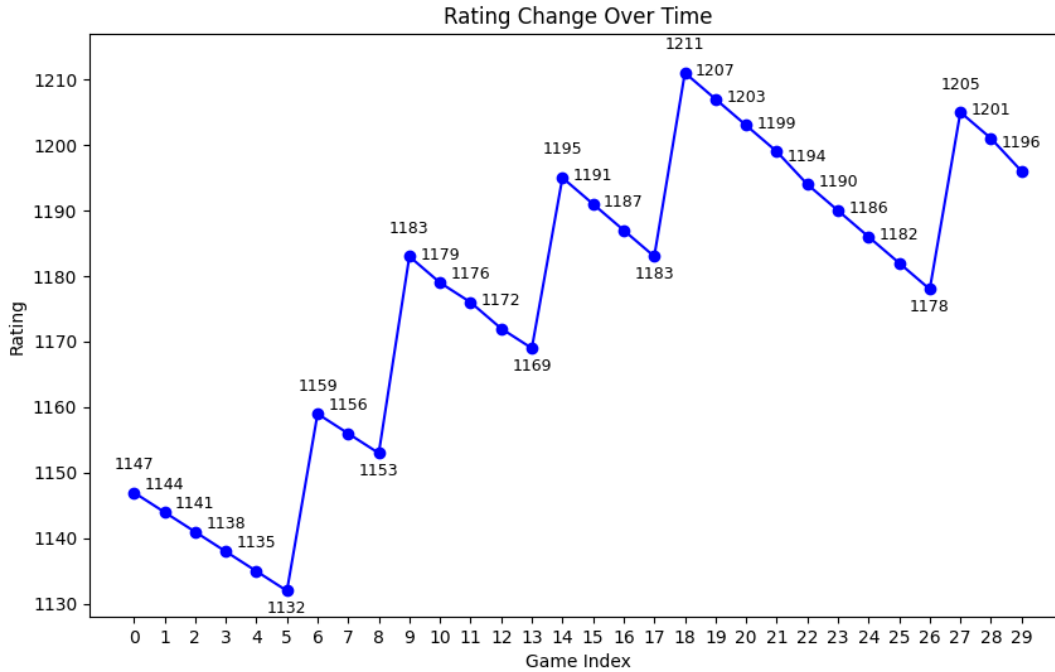


Figure 5: graph of rating over games

The figure 5 shows the change of rating in the span of 30 games. Only 30 out of 40 games are shown on the graph since the first 10 games on colonist website in ranked game mode are called placement matches. That means they are used to asses the starting rating of a player and do not give an immediate rating. As can be seen on the figure all wins out of the 40 games happened after the placement games. This is probably due to being placed against similarly ranked opponents instead of better players assigned during placement games. The rating seems to have stabilized slightly below 1200. The mean fall above the rating of 1180 is 4.5 and the gain after a win is 27. This means that for 6 lost games 1 should be won to maintain the rating. That is the win rate to maintain the rating is 14.3%. It is higher then the overall win rate of the player. However, after excluding all lost placement games played against better opponents the win rate is equal to 16,7% which is slightly higher then the one needed to maintain rating. This means that the agent could probably climb slightly in rating.

Agents final rating is not great. Multiple downsides of the value function and the methodology crystallized themselves after analyzing the games. The most important are, not enough depth during initial settlement placement, over valuating having production of all types of resources at the start and overrating open settlement spots.



(a) agents initial placements



(b) final board state

Figure 6: initial and final board state in one of the worse games by the agent

The board in figure 6 is a good example of what happens when all of those mistakes present themselves in one game. The agent in this game is placing in the forth position that is, it places both settlements one after another. Normally this is a strong position since multiple opponents have already placed their starting settlements which allows for creation of more predictable game plan. This is even more true for low ranking games were players tend to make unexpected starting placements. However, the agent is not able to take advantage of this. Firstly, it is quite obvious that one of the two intersections with surrounding tiles with numbers 8, 3, 4 should be taken. They are the highest producing fields. The Agent chooses the one on the left side of the board since it evaluates highly the possible expansion to the intersection of tiles 10, 3 and 11. What the agent is not able to take into consideration is that blue player will gladly place their settlement

on the intersection of 6,3, 11 blocking the previously mentioned expansion. Getting to high depth during initial placements is challenging since there are around 40 possible placements for the first settlement, for each there are 3 road possibilities, later there are close to 37 possible spots for the second settlements and once again 3 roads for each. This gives 13320 combinations before considering any future opponent placements and is already higher then the limit of 10000 leaf nodes set to assure that there is enough decision time left for evaluation and execution of the action. After the first placement on the 8, 3, 4 there is only one spot on the board left which provides both resources that the player lacks, namely 6, 11, 12 which the agent chooses. This propensity to choose produce all 5 types of resources at the start is a carry over from a one on one play stile where opponents are less likely to trade and there is more space on the board due to there being less players. As can be seen on the final board, low production in combination with 2 starting road that ended up being of no use because of blues starting placement led to a swift defeat. Nevertheless, even though the value function and search methodology are imperfect winning multiple games using hand made weights and metrics converted from a different game mode is positively surprising

4.2 Alternative model

This model always performs the alternative action chosen by the process described above. The alpha parameter was chosen to be 1.5. This allows to differentiate between different road placements but does not consider asking someone for skipping their turn instead of building something. When testing with hard bots on colonist website across 10 games it performed 2.6 alternative moves per game. This bot in 40 ranked games reached the final rating of 1234 points, it had a win rate of 15% and collected on average 6.36 points per game.

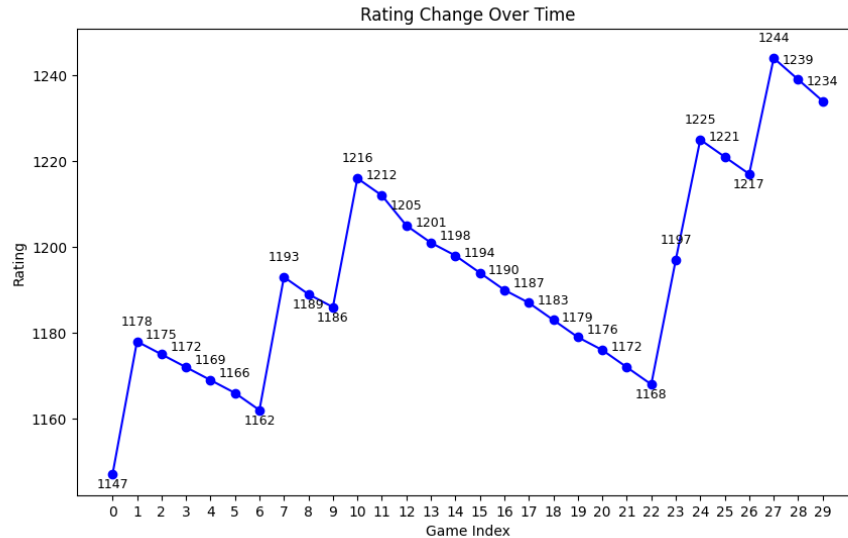


Figure 7: graph of rating over games

The rating graph above shows the progress of rating across 30 games starting from assigned rating

after the placement games. As can be seen on the graph the software was assigned the same starting rating. However, this version managed to win 3 times while playing the opponents on its level reaching a higher rating in the end.

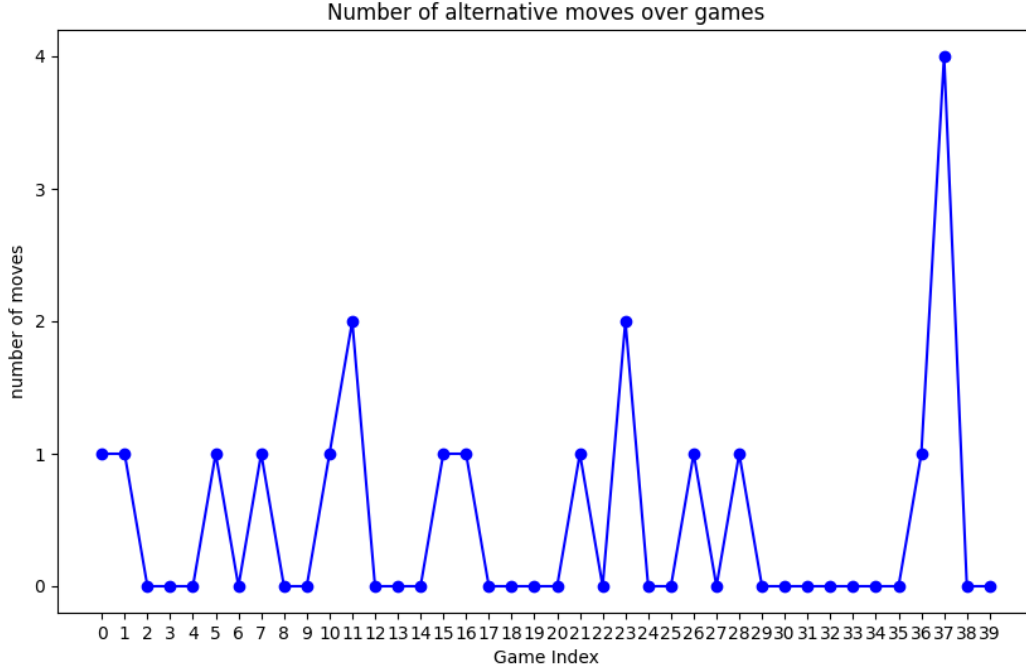


Figure 8: graph of number of alternative moves performed per game

The figure 8 shows the number of alternative moves performed in each ranked game where index 10 is the 0 in figure 7. As can be seen the average number of alternative moves performed per game, equal to 0.475, is a lot smaller then the number performed against colonist bot which was 2.6. The reason might be decreased congestion on the board, since players tend to place starting settlements further apart from each other then bots. Interestingly all won games included at least one alternative move, the average number of points is also substantially bigger, 0.7 points per game more then the base model. This might be a side effect of most alternative moves, 72%, being ending turn. The reason for this behavior is the nature of the type of search performed. If the turn of the agent is ended without performing a move, it leaves more depth to explore options of opponents therefore increasing the chance for an alternative move being found.

It is probable that ending the turn in such a scenario leads to solving one of the issues mentioned before, namely unwillingness to hold resources to build settlements. In a scenario where alternative moves are not considered, not only next turn of the agent is rarely reached but also when it is, its value is diluted due to the stochastic nature of the action of rolling dice. By calculating the expected value of 4 rolls of dice in a row, one for each player after ending the turn, if in only few of these scenarios the agent is able to build a settlement its value when backtracked to the agents end turn action might be smaller then building a borderline useless road. When introducing the alternative move the increase in value does not stem from agents better move in the next turn

but from worse move by the opponent which happens shallower in the search tree. This decreased depth of states with grater value leads to it backtracking through less expected value calculations of stochastic actions. That in turn leads to the value of ending turn increasing more then before, which makes the agent hold its resources more often allowing for a chance to get resources for a better move.

4.3 Conversation model

The conversational model sends in game messages trying to ask players to make a suboptimal move as described in the methodology section. After playing 22 games it reached a win rate of 9.1%, a rating of 1209 and collected on average 5.5 points.

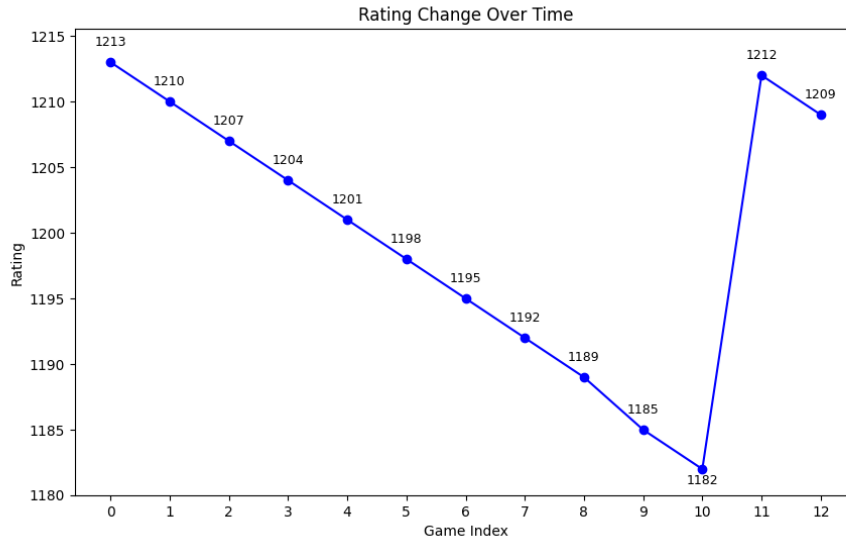




Figure 9: graph of rating across games

As can be seen on the figure 9 the agents starts with a higher rating due to winning one placement game and proceeds to loose most of the other games.

The win rate of the agent is incredibly small. The bot never performed the alternative action since other players either did not come to an agreement or did not respond to the message within allocated time window of 10 seconds. However there are other factors that might have contributed to the low win rate.

Ender2928: Woo1189, would building a road to the intersection of desert, , and  be beneficial for you?
Woo1189: Dont know, haven't thought that far

(a) Conversation between a player and the agent

Woo1189: Let's team up red

(b) attempt to team up with the agent

Figure 10: In game chat screenshots showing an example of conversation, and overestimation of the abilities of the agent.

Figure 10 first shows a bot, Ender2928, starting a conversation with a player called Woo1189. It tries to convince it to build a road to a suboptimal spot next to a desert which produces no resources. The player responds after more then 10 seconds therefore no further attempt of convincing them took place, however it had an interesting side effect. Woo1189 suggests teaming up with the agent. This illustrates that the presence in chat might increase perceived capabilities of the agent leading to additional actions taken against it by the opponents and lowering the win rate.

When a game happens when there are many alternative moves possible, like the index 37 in figure 8, other downsides of the conversation model become visible.

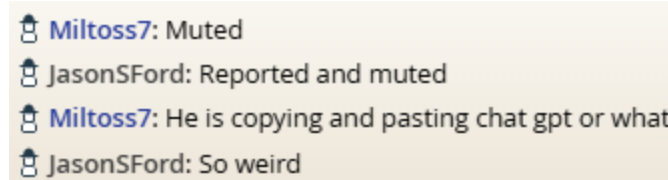


Figure 11: screenshot from in game chat showing annoyance

The disconnection between ongoing conversation and previous messages from the agent when coincided with the bigger amount of messages led to suspicions growing in the opponents. In addition the LLM style of messages, due to the model not being fine tuned in the end, gave additional reasons to come to the conclusions shown in the figure 11. Those might not have been the only annoyed opponents by the agent during the course of its games which could also contribute to its lesser performance. The game from which the above messages were taken was the last one played by this model due to an apprehension that it may lead to further inquiry in not only this agent but also others and endanger the experiment, even though running bots is not strictly against the policy of colonist.

In conclusion the attempt at improving the model by using in game chat was not successful. The attempts at communication increased the perceived threat in the agent and made the opponents play against it. To achieve better results firstly a substantially better value function should be created. Although, as shown by the agent performing alternative moves, the core idea of finding suboptimal moves for opponents seems promising.

5 Conclusions

An underestimated amount of work went into extracting the information from the colonist website. The procedure includes many computer vision technics like template matching and optical character recognition. This in turn led to imperfections in other regions of the project. The value function on its own is an interesting field of study and could be substantially improved. The communication procedure went through many iterations, testing 3 different base large language models with different levels of finetuning. The mistral model proved very flexible in its use cases and consistent in its quality of messages although a fine tuning process could help it better blend in with other players during the game. The core idea of searching for spots suitable for exploitation held true, as the player making the alternative moves had higher final rating, higher win rate and higher points per game then the base model. However the conversational model did worse then expected. The presence

in chat led to bigger perceived thread and the unnatural style of writing led to miscommunications and annoyance which in turn led to a worse performance in terms of win rate and average points. The only positive metric, rating, is only this high due to winning one of the placement games which underlines the importance of repeating the experiments multiple time which was not feasible during this project. In conclusion the attempt at improving the performance of catan agents using natural language was not successful, however, many insights into the topic were gained and the idea of exploiting the opponents mistakes seems to be a promising field for future studies.

6 Further research

There are multiple improvements that can be made. Two greatest possible improvements are adding player trades and better value function. It has been previously shown that conversation can impact between player trades [3] and it could be further improved using large language models. It could not be implemented here due to limitations posed by data extraction from colonist user interface. The trades have a quick timer that would not allow for proper evaluation, they often appear in irregular fashion with multiple trades at a time and if there are more then three opened trades at once they have to be scrolled through. All of the aforementioned roadblocks would not be impassable if a better way of extracting information existed. When it comes to better value function it is a promising research path. Many improvements can be made to the one used in this thesis, one way is including more enemy metrics like number of development cards on their hand or how easily can they be blocked and then optimizing the weights. The other could be to develop a neural network for evaluating the position. This approach had great successes in other games, and some in catan [6], but ended beyond the scope of this thesis due to time taken to develop feature extraction methods from colonist. Another improvement that could be made is in the methodology used to describe board state to the large language model. Including special tokens, similarly to how it was done for diplomacy agents, [2] [9], would lead to richer understanding of the position and therefore more convincing messages.

References

- [1] Available at <https://github.com/bcollazo/catanatron>.
- [2] Meta Fundamental AI Research Diplomacy Team (FAIR)[†], Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of `ji:diplomacy:i:` by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.
- [3] Markus Guhe and Alex Lascarides. The effectiveness of persuasion in the settlers of catan. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8, 2014.
- [4] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile

Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mistral 7b, 2023.

- [5] Ollama Team. Ollama. Available at <https://ollama.com>.
- [6] Michael Pfeiffer. Reinforcement learning of strategies for settlers of catan. In Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, 2004.
- [7] Colonist team. colonist. Available at <https://colonist.io>.
- [8] Qwen Team. Qwen2.5: A party of foundation models, September 2024.
- [9] Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. Diplm: Fine-tuning llm for strategic decision-making in diplomacy, 2025.
- [10] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. arXiv preprint arXiv:2407.10671, 2024.