Transformer Model Game AI: Preliminary Results

Jonathan Stokes

June 10, 2024

1 Introduction

Large Language Models (LLM's) are currently seeing success in many AI applications including as AI for games. Initially I considered fine tuning LLama2 or another LLM to create a game AI. However since I have limits in terms of available compute and memory I decided to scale back my goals and instead create a game AI using a Transformer model.

Having read the Transformer paper "Attention Is All You Need" [4] but not having either the time or the deep understanding required to implement the model from scratch I opted to adapt an example in PyTorch's documentation [3]. This example uses a Transformer model to translate German to English. My thinking in choosing to adapt this example was, what is a game AI but do except translate from a the state space of a game to an action?

2 GOPS

The game I selected to train an AI for is the Game of Pure Strategy (GOPS) or Goofspiel. Two player GOPS is played with a standard deck of 52 cards. The one suite is shuffled and placed face down on the table, these are the prizes. Two of the remaining suites are selected and given one each to players as their hands. A round of GOPS starts when the top card of the prize deck is flipped face up displaying its value (Ace being low and King high). The two players then secretly select cards from their hands to bid for the prize. These cards are placed face down on the table. The round ends with both players simultaneously flipping over their bid cards with the bid value being the card's value. The player with the highest bid wins the rounds prize, if the bids tie the prize card remains on the table and is added to the next rounds prize. The cards used to bid for the prize are discarded and the next round begins. Play continues until the prize deck is exhausted at which point each player calculates their score as the total value of the prize cards they have won and the player with the highest score wins the game.

3 Modeling GOPS's State Space

It is a easy to compute the number of states in GOPS since there are 13! ways for the prizes cards to be ordered and 13! ways for each player to play their bids giving a total of

$$(13!)^3 \approx 2.41 \times 10^{29} \tag{1}$$

game states. This is far more then the number of grains of sand on earth, roughly 4×10^{20} [2], but far less than the number of chess states, roughly 4.82×10^{44} [1].

One option for modeling the state space for GOPS is to map every state to a unique integer, so for instance the initial state s_0 for GOPS might be mapped to integer 0. The goal would then be for the Transformer

model to learn a mapping from integers to an action a in the set of actions $\mathcal{A}$ with the highest probability of resulting in a win for the AI agent,

$$a_i = \underset{a_i \in \mathcal{A}_i}{\text{argmax}}(p_{win}(a_i|s_i)) \tag{2}$$

where i is the integer the state is mapped to and $\mathcal{A}_i$ is the set of legal actions in that state.

However I thought there might be two major issues with modeling GOPS state space this way. First, in modeling GOPS state space as $(13!)^3$ integers we are assuming that the sequence of cards is important GOPS. However clearly the sequence in which cards appear is not always important. For instance consider game state s_a where,

Turn	Prize	Player 1 Bid	Player 1 Bid
1	Ace	Ace	Ace
2	Two	Two	Two

Table 1: Sequence of cards producing s_a

and game state s_b where,

Turn	Prize	Player 1 Bid	Player 1 Bid
1	Two	Two	Two
2	Ace	Ace	Ace

Table 2: Sequence of cards producing s_b

The properties of states s_a and s_b are the same but if we model GOPS state space using the integer mapping above $s_a \neq s_b$ which implies that some of the states which result from the integer mapping are redundant. In other words, our model of GOPS's state space should take advantage of the fact that if we are not attempting to infer our opponents strategy, GOPS obeys the Markov property.

Second, the strength of Transformer models is in their ability to attend to different parts of an input sequence. If the input sequence is a single integer I would guess (although I have yet to test this) that the Transformer model is under utilized and perhaps another model or even a lookup table would outperform it.

Given these consideration I chose instead to represent the game state of GOPS as a structured statement consisting of the following word types:

player_hand
opponent_played_cards
player_score
opponent_score
prize_card
previous_prize_cards

Table 3: GOPS word types

I cannot prove that a statement of this form is an optimal representation for GOPS states. But under this representation $s_a = s_b$ eliminating some of the states that appeared redundant in the integer mapping and the Transformer model could in theory learn to attend more to the cards in its hand than its opponents score in choosing a card to play or vice versa.

4 Training Data

To generate the training data I let two uniformly random AI agents play each other $10k$ times. These agents selected their bids with equal probability from the cards in their hands. I used the approximately $130k$ state action pairs from these games that resulted in wins to train the Transformer model. Here the approximation is due to moves from tied games not being included in the training data. I then repeated this processes but had the Transformer agent AI play against a uniformly random agent to generate the next set of training data, repeating this process of "Bootstrap Training" until I performance of the Transformer agent began to degrade. I do not have a good explanation for why the models performance began to degrade after several iterations of bootstrap training, except that it might over fit the training data or in reinforcement learning terms that it stops exploring actions.

5 Model Parameters

For the Transformer model implementation I used PyTorch's Transformer class and modified the embedding functions used in PyTorch's translation example [3] for the GOPS state space statements. However given that the state space of GOPS while large is much smaller than the state space of all German sentences and that the set of all GOPS actions is 13 as opposed to all English sentences, I reduced that capacity of the Transformer model in a bid to reduce the likelihood of the Transformer model over fitting the training data. There parameters I settled on are listed below.

Parameters	Values
Attention heads	4
Batch Size	1024
Decoder Layers	2
Dimension of FFN	256
Dropout	0.1
Embedding Size	128
Encoder Layers	2
Loss Function	Cross Entropy Loss
Optimizer	Adam
Optimizer betas	$(0.9, 0.98)$
Optimizer eps	1×10^{-9}
Optimizer learning rate	0.0001

Table 4: Transformer Model Parameters

In training I used 10% of the data as a validation dataset and stopped training the models transformer models the epoch before the loss increased on the validation data. Using a validation dataset to evaluate over fitting is not ideal as it is only a proxy for model performance in playing GOPS, however it is straight forward to implement.

6 Results

First lets point out some issues with these results. As I was generating these results in the course of developing the model I did not record the results for or necessarily run all the same number of trials for each experiment. This is why I call these preliminary results, but with that caveat I think these results pass the sanity test. In Table 5 and Figure 1 we see that the Transformer model AI's highest win rate 0.64 which is achieved on the third iteration of bootstrap training. After the 4th iteration of bootstrap training the Transformers model AI's win rate decreases to 0.62.

Iteration	Wins	Trials	Win Rate	Experiment Name
0	508	1000	0.51	rand_v_rand
1	580	1000	0.58	rand_v_transformer_1
2	6144	10000	0.61	rand_v_transformer_2
3	6440	10000	**0.64**	rand_v_transformer_3
4	620	1000	0.62	rand_v_transformer_4

Table 5: Transformer model AI wins and win rate against a uniformly random AI.

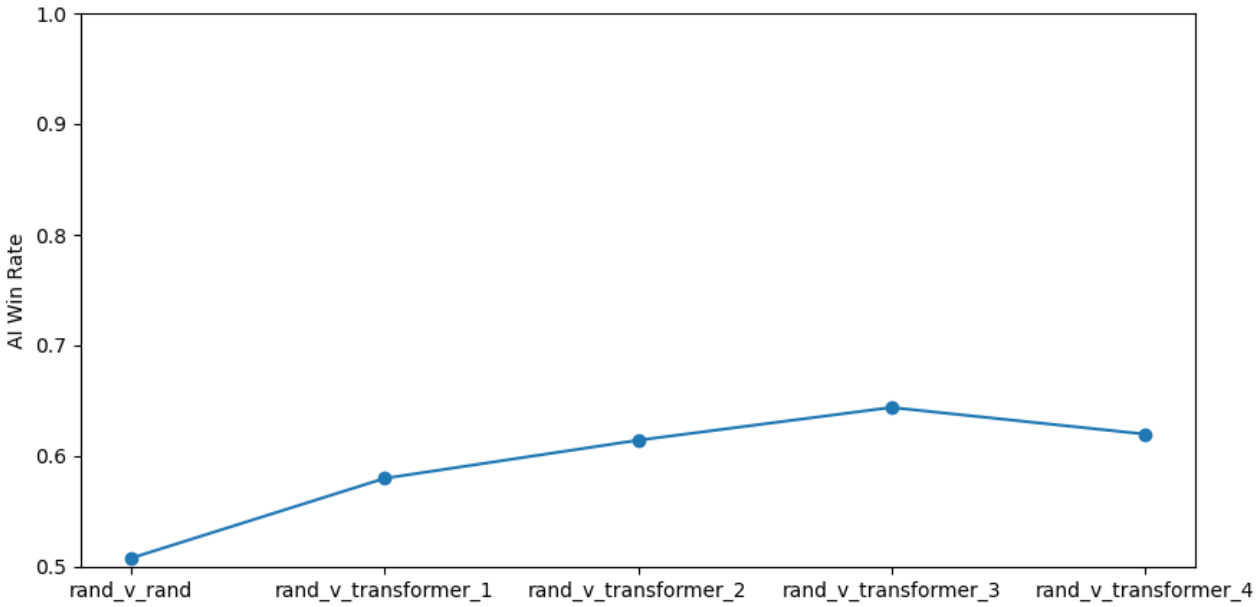

Figure 1: Transformer model AI win rate as a function of bootstrap training iteration.

To get a sense of how training the Transformer model against a uniformly random opponent effects the distribution of the Transformer models actions see Figure 2 and Figure 3. These figures show the number of times a card was played as a function of the prize value. Prize value is only one dimension of GOPS's state space but it is arguably the most important dimension and these figures show that Transformer_3 appears to be more likely to play low cards if the prize is low and high cards when the prize is high compared to a uniformly random agent.

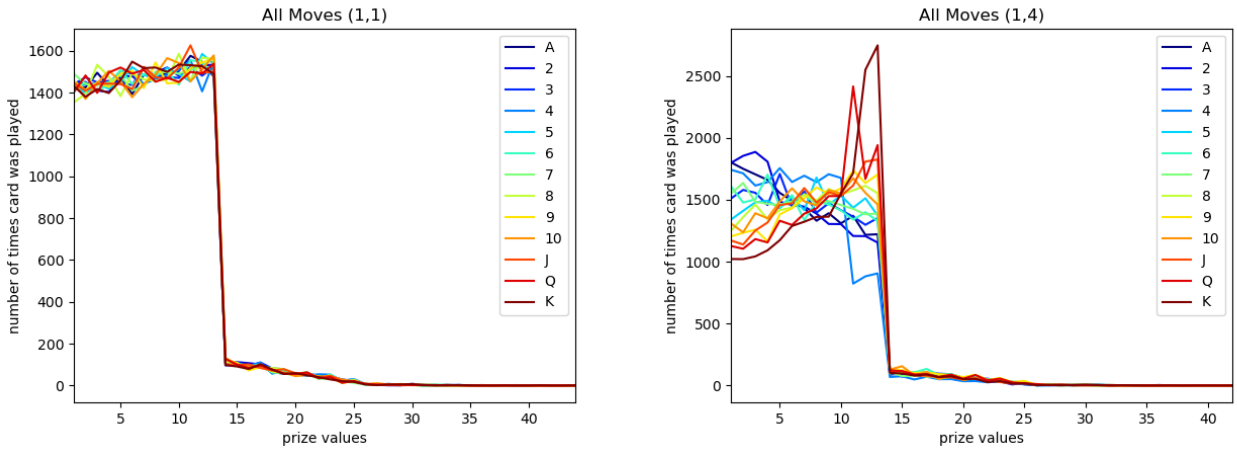

Figure 2: Number of times a card was played as a function of prize value for all played cards. **Left:** random AI vs random AI. **Right:** random AI vs transformer_3

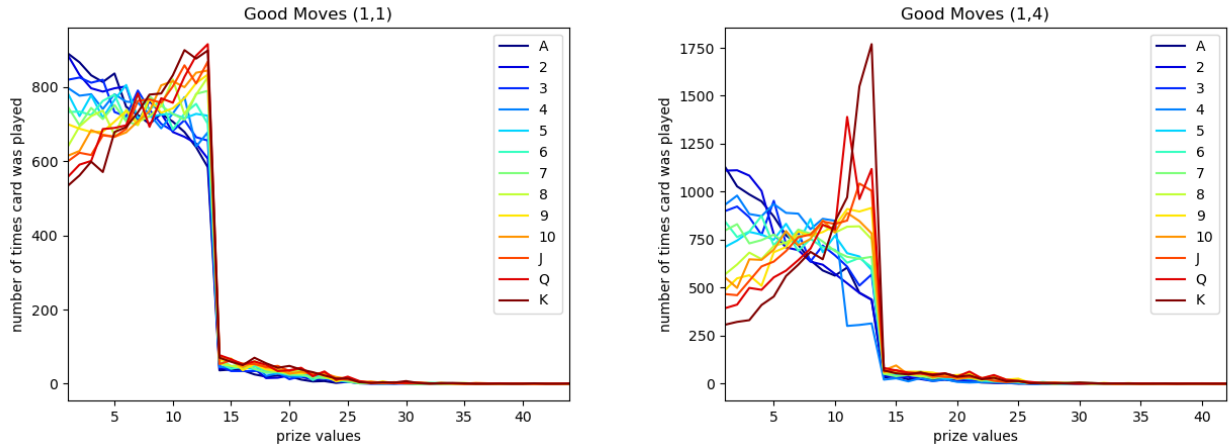

Figure 3: Number of times a card was played as a function of prize value for played cards that resulted in wins. **Left:** random AI vs random AI. **Right:** random AI vs transformer_3

7 Conclusion

While I think this work shows that Transformer models can be trained for game AI applications, it also shows given the relative weak performance of the resulting AI agent against a random opponent that there is a lot of work that could be done to improve Transformer based AI agent performance. For instance, would more carefully selected training data produce better performance and avoid the decline in performance after repeated iterations of bootstrap training?

Transformer_3 can be downloaded and tested in an implementation of GOPS at:

`https://github.com/sophist0/gops`

References

[1] Chesspositionranking. `https://github.com/tromp/ChessPositionRanking`. Accessed: 2024-06-09.

[2] Do stars outnumber the sands of earth's beaches? `https://www.scientificamerican.com/article/do-stars-outnumber-the-sands-of-earths-beaches/`. Accessed: 2024-06-09.

[3] Language translation with nn.transformer and torchtext. https://pytorch.org/tutorials/beginner/translation_transformer.html. Accessed: 2024-06-09.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.