IANNwTF - Winter term 2022/23
GPT-2 model learns the rules of chess

Clive Tinashe Marimo (cmarimo@uni-osnabrueck.de)
Moritz Lönker (mloenker@uni-osnabrueck.de)
Silvie Opolka (sopolka@uni-osnabrueck.de)

supervised by Mathis Pink (mpink@uni-osnabrueck.de)

April 1st 2023

Context and motivation

Chess is a complex game that requires a deep understanding of strategy and tactics. According to DeLeo et al. (2022), chess has around 10^{43} unique chess board states, as well as a branching factor of approximately 35, i.e. in most game positions, there are about 35 possible moves that can be made. Additionally, Shannon estimated that there exist 10^{120} unique chess games.[1] This makes chess a computationally challenging problem, even for today's advanced models and capacities.

In the past years, multiple strong chess engines (e.g. AlphaZero and Stockfish) got published that are capable of beating the most advanced human chess players.[2]

However, there is rather little research on using natural language processing (NLP) to train a language model to play chess from chess transcripts exclusively. The objective of this project is to train a Generative Pretrained Transformer (GPT) model to learn how to play chess from chess transcripts (i.e. a series of moves) of several human plays, without having an explicit representation of the game board state and game rules. We want to investigate, if it is possible for such a model to 'learn' and apply the complex rules and theory of chess, much like it is possible for established transformer models to learn and apply complex grammatical rules.

This project has the potential to lead to new insights and understanding of how language models can learn and understand complex concepts. Additionally, it will demonstrate the diverse and powerful nature of transformer models once again.

Related literature/research

In 2019, DeepMind introduced the algorithm MuZero, which is able to master games, including chess, without knowing their rules.[3] In the following years, more research projects attempted to train a language model to play chess without encoding the rules of chess explicitly.

DeLeo et al. (2022) applied a BERT model to the game of chess, using a game state encoding in Forsyth-Edwards Notation (FEN), and a move encoding in coordinate algebraic notation. After training the model for three days on a dataset covering only chess game openings (i.e. containing only the first three moves per game), the model was able to generate 75 percent valid moves at 35 moves (per player) into the game. This observation shows that a language model is able to generate chess games, even with very little information, given it is fed with a large enough amount of data.

Stöckl et al. (2021) analyzed the effect of GPT-2 model size (small, medium, large), training time (0 - over 120 train hours), and amount of training data (99604, 577202, 2163417 games) on the model's learning success. For evaluation, the models were saved at multiple training time steps.

In this study, the following important observations were made:

First, the usual evaluation metrics for language models (accuracy, perplexity) seem to give no reliable indication of the model's performance success. Results show that both, accuracy and perplexity measures, do not vary (significantly) between different model sizes, suggesting that smaller models can learn the rules of chess almost similarly well/quickly as larger models. However, chess-specific metrics show a clear difference between the model sizes. Additionally, both, accuracy and perplexity measures, indicate a strongly decreased learning rate when increasing the amount of training data. However, chess-specific metrics show a significant increase in learning progress when using larger amounts of training data. Based on those findings, we will use chess-specific metrics for our model evaluation.

Second, the raw pre-trained GPT-2 model was able to generate look-alike chess games. However, those hardly contained legal moves, independent of the model size. This bad performance is explained by the insufficient and inconsistent chess training data: Insufficient because the model was trained more generally, not specifically on chess data. Inconsistent, because the training data might have contained multiple chess notation variants. Those findings highlight the importance of fine-tuning the model on chess data and the need of adjusting its inner architecture where necessary.

Third, they could observe that the performance of the model improves, the more time is invested in training. Their graphical visualizations show a strong but increasingly limiting improvement. The strongest improvement appeared in the first 20 hours of training. This gives us an insight into the training time needed to observe sufficient results.

[1] https://github.com/ricsonc/transformers-play-chess

[2] https://www.chess.com/terms/chess-engine

[3] https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules

Fourth, the medium-sized model with medium data volume seems to be the best-performing option, providing a good trade-off between performance capabilities and computational resources. After 20 train hours, this model generates an average sequence of about 28 legal moves (before generating the first illegal move) from a known opening position, an average sequence of 11 legal moves from a game position after ten moves, and an average sequence of 2.8 legal moves from a game position after ten random moves. Confirming the results of Kaplan et al. (2020), their observations suggest that model size and data volume should be increased together for good results.

Noever et al. (2020) fine-tuned a GPT-2 architecture to learn the rules of chess, using a rather large data volume with 2.8 million chess games in Portable Game Notation (PGN). Their evaluation focused on special PGN tokens (castling, pawn promotion, check, checkmate) and common opening moves (like ”English Opening” and ”Slav Exchange”). After 30,000 training steps, the model was able to generate chess games with 10 percent illegal moves. Common moves were king-side castling (17.74 percent in 1000 generated games) and pawn promotion (13.67 percent in 1000 generated games).

Oshingbesan et al. (2022) conducted research in the field of multi-domain, multi-task learning. Their research goal was to train a small GPT-2 model on code and chess data, such that it performs well in four different tasks (Chess move generation, Chessboard state evaluation, Code generation from an English prompt, and Code summarization using the English language) of the two unrelated domains “Python code” and “chess”. When comparing several popular training strategies, the model resulting from GPT-style joint pre-training and joint fine-tuning was able to keep multi-domain knowledge the best, while still performing reasonably well in the individual tasks. Their baseline GPT-2 model is smaller compared to the model used by Noever et al. (2020). However, the model performance (9.6 percent illegal moves) does not differ significantly, suggesting that even smaller models can perform equally well when using more training steps (50,000 instead of 30,000) and training data (14.3 million games instead of 2.8 million games). Another crucial observation is that all tested models struggled to end the chess game.

Presser et al. (2020) made a similar observation, where the model started to blunder around move 13. Like Oshingbesan et al. (2022), they suspect this is due to losing track of the board state. In this research project, we attempt to reduce this problem.

Common chess notations

During our literature review, three different notations appeared frequently:

1. **Portable Game Notation (PGN)** contains metadata (e.g. names of players and their ELO ratings, date and time of the game, result) and the game transcription (i.e. the sequence of moves) in Standard Algebraic Notation (SAN). It represents moves without explicitly stating the start position of the piece being moved (if there is no ambiguity). [4] [5]

2. **Long algebraic notation (LAN)** represents moves using coordinates of the start and end position of the piece being moved (e.g. c2-c4). [6]

3. **Forsyth-Edwards Notation (FEN)** describes a particular board state by representing the position of each piece on the board, including empty fields. [7]

In this research project, we use all three notations.

The dataset

For training, testing, and evaluation, we use data coming from chess transcripts. The chess transcripts are downloaded from the internet chess server Lichess, which provides an open-source database containing all games played on the server since 2013.[8] Currently (March 10, 2023), there are 1.3 TB available with information on over 4 billion (standard) chess games. The chess games played on Lichess might not be representative of all types

[4] https://chesspedia.fandom.com/wiki/Chess_Notation

[5] https://support.chess.com/article/658-what-are-pgn-fen

[6] See footnote 4

[7] See footnote 5

[8] https://database.lichess.org/

of games and players, but they provide enough information about the rules of chess. Therefore, collecting data from multiple sources, to cover as much diversity as possible, is not necessary.
The transcripts come in .pgn.zst format. PeaZip[9] was used to decompress the data to get the PGN files.
We did not include chess variants (e.g. Antichess), but only used standard chess game data. Considering the reported issues on Lichess, we avoided using data that is known for at least one of the following issues:

- games with illegal castling moves
- games with missing black move numbers after comments
- games that were resigned after game end
- games with incorrect results

The model

Huggingface provides many pre-trained GPT-2 models.[10] We chose to fine-tune the *GPT2LMHeadModel*, which is a GPT-2 model with a language modeling head. [...]

The methods

how exactly are you solving the problem at hand, how and why does everything work? Consider visualizing your architecture and if appropriate important structures from your methods.

The implementation

How does your implementation work? Try to explain your implementation choices and walk the reader through your implementation. Consider making use of pseudocode, appropriate visualizations (e.g. for explaining the structure) or even detailing crucial code elements in detail.

Data Preprocessing

[...]

Model class

[...]

Training

[...]

Evaluation

[...]

The results

How did your experiments turn out? Of course training (typically average return vs. training samples) should generally be included, but try to give more details: What exactly happened, can you create an additional experiment to explain what causes your results? Which part of your method causes the results (if you combine multiple approaches?). If appropriate, include a qualitative analysis of your results. Make sure to use appropriate visualization for this section!

[9] https://peazip.github.io/
[10] https://huggingface.co/docs/transformers/model_doc/gpt2#transformers.GPT2LMHeadModel

Discussion

Put your results and overall project into context: What worked, what did not? What should further be improved? What follow-up questions would you like to ask?

Follow-up questions

1. Beginners and advanced players differ in the moves that they use. Do the average ”goodness” and range of “goodness” of training games have an effect on the model’s success in generating legal moves? The ”goodness” of a chess game can be inferred from the move evaluations (e.g. by Stockfish[11]) and ELO ratings of the players[12] that are calculated based on their past performance.

[...]

Sources

DeLeo, M., and Guven E. (2022). Learning Chess and NIM with Transformers. In International Journal on Natural Language Computing. DOI: 10.5121/ijnlc.2022.11501

Kaplan, J., McCandlish, S., Henighan, T., Brown, T., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.

Noever, D., Ciolino, M., and Kalin, J. (2020). The chess transformer: Mastering play using generative language models. arXiv preprint arXiv:2008.04057.

Oshingbesan, A., Ekoh, C., Atakpa, G., and Byarugaba Y. (2022). Extreme Multi-Domain, Multi-Task Learning With Unified Text-to-Text Transfer Transformers. arXiv:2209.10106v1 [cs.CL].

Presser, S., Branwen, G. (2020). A Very Unlikely Chess Game. slatestarcodex.com/2020/01/06/a-very-unlikely-chess-game/.

Stöckl, A. (2021). Watching a Language Model Learning Chess. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), pages 1369–1379, Held Online. INCOMA Ltd. doi: https://doi.org/10.26615/978-954-452-072-4_153

[11] https://stockfishchess.org/

[12] https://www.chess.com/terms/elo-rating-chess