UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Raiko Marrandi

Evaluating Transformer Architecture for the Game of Chess

Bachelor's Thesis (9 ECTS)

Supervisor(s): PhD Eduard Barbu

Tartu 2023

Evaluating Transformer Architecture for the Game of Chess

Abstract:

Transformers are state-of-the-art natural language processing models, which have shown success in a variety of areas not directly related to natural language. This work evaluates the learning capabilities of transformers in the game of chess. The models are trained using an unannotated dataset of played chess games in Forsyth-Edwards notation (FEN) and their performances are compared with models trained on less comprehensive datasets used in prior research. The findings show that the models are not capable of generalizing on the richer FEN dataset and demonstrate inferior performance compared to the control models across all evaluation metrics.

Keywords:

Machine learning, transformers, chess, self-attention, unsupervised learning

CERCS: P176 Artificial intelligence

Transformer-arhitektuuri hindamine males

Lühikokkuvõte:

Transformerid on tipptasemel loomuliku keele töötlemise mudelid, mida on efektiivselt kasutatud ka ülesannetes, mis ei ole otseselt seotud keeletehnoloogiaga. See töö hindab transformerite õppimisvõimekust males. Mudeleid treenitakse kasutades ilma märkusteta andmestikku male partiidest, mis on Forsyth-Edwards notatsioonis (FEN). Nende tulemuslikkust võrreldakse mudelitega, mis on treenitud varasemates uuringutes välja pakutud lihtsamate andmestikega. Töö tulemusena leiti, et transformer-mudelid ei ole võimelised kompleksemal FEN andmestikul üldistama ning need näitavad kõigil mõõdukitel nõrgemaid tulemusi võrreldes kontrollmudelitega.

Võtmesõnad:

Masinõpe, transformerid, male, enesetähelepanu, juhendamata õpe

CERCS: P176 Tehisintellekt

Table of Contents

Introduction

Self-attention-based transformer architecture (Vaswani, et al., 2017) has become the *de-facto* model of choice in natural language processing (NLP). It has also proven to be successful as the foundation of large language models in a variety of areas not directly related to natural language, like image recognition (Dosovitskiy, et al., 2021), predicting protein structures (Jumper, et al., 2021) and music generation (Dhariwal, et al., 2020).

Transformer-based models have been shown to be capable of learning arithmetic, a task that requires rule-based reasoning, to a certain extent with only training data crawled from websites (Brown, et al., 2020). While Nogueira et al. (2021) demonstrated that transformer-based language models are incapable of learning rules of arithmetic beyond the length of numbers seen during training, which implies inherent limitations arising from the lack of information in training data.

The thesis evaluates, whether state-of-the-art language models are capable of learning the rules of chess based only on records of previously played games. In addition to historical relevance, the field is also an opportune testing area thanks to large databases of publicly available data on online chess servers and evaluation methods based on rules, in addition to usual language model evaluation metrics.

NLP methods have been applied previously to the game of chess to research the rule-learning mechanisms of the architecture (Noever, Ciolino, & Kalin, 2020), (Stöckl, 2021). This thesis explores the problem from a similar perspective, using only unannotated transcripts of games to train the models and creating a system that discovers the rules of chess by itself, while improving on the methodology of previous research with a richer data format. Finally, the work of Stöckl (2021) is replicated and the results are compared to the new model.

The work is divided into four chapters. The first chapter discusses the background of the thesis, related works, and the formulation of the problem. The second chapter describes the data and pre-processing methods. Chapter three details the training and evaluation methods and chapter four outlines the results, comparing them to the replicated implementation of Stöckl (2021).

1 Background

This work is based on artificial neural networks (ANNs), the origin of which dates back to the 1940s, when the neurophysiologist Warren McCulloch and logician Walter Pitts (1943) proposed the first computational model of a neuron. It is now referred to as the McCulloch-Pitts neuron. Their neuron had hard limitations though, it had fixed weights, which meant it couldn't learn from data, and it only worked with boolean values.

A major improvement to their neuron was proposed by Rosenblatt (1958), called the perceptron. Rosenblatt's perceptron could, compared to the McCulloch-Pitts neuron, work also with numerical values and learn its weights from data, using the first supervised learning algorithm, which was aptly called "the perceptron learning rule". Yet it was still limited to simple, linear decision boundaries and would come to be replaced by ANNs (Sinai, 2017).

1.1 Machine Learning

The three major categories of machine learning (ML) are supervised learning, unsupervised learning, and reinforcement learning (RL). Of these three, supervised learning has been around the longest (Rosenblatt, 1958), its main characteristic being, that it uses labelled datasets to train algorithms to either classify data or predict outcomes based on data (IBM, 2023). Unsupervised learning, on the other hand, works to discover patterns and groupings by analysing and clustering unlabelled datasets, so it does not require human intervention nor labelling (IBM, What is Unsupervised Learning?, 2023).

Reinforcement learning methods differ from the other two. These enable an agent to learn through trial and error by freely interacting with its environment and maximizing the total cumulative reward from feedback, in the form of rewards and punishments, on its actions (Bhatt, 2018).

While these three are the major techniques in ML, in practice they're rarely used alone and are instead, sometimes inherently, combined with other methods. The form of machine learning used in this project can be called deep unsupervised transfer learning, since a pre-trained deep neural network is fine-tuned to solve a new task on unannotated data.

1.2 Transformer Architecture

Transformers (Vaswani, et al., 2017), a type of deep neural network, are based on the mechanism of self-attention, which allows the language model to focus on relevant parts of the input sequence in order to compute a representation of the sequence. Self-attention is implemented as a scaled dot-product function using query, key, and value matrices generated from the embedding of each input token. The output is then computed using the dot product of query and key, divided by the square root of the key matrices' dimension, followed by *softmax* normalization and multiplication with the value matrix, see (1). Instead of a single attention function though, the paper further details multi-head attention, which allows the model to attend to information from different representation subspaces at different positions all at once. Multi-head attention achieves this by linearly projecting all matrices h times with different, learned projections to their respective dimensions and performing the attention function in parallel (Vaswani, et al., 2017 / ibid.).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right)V \ . \quad (1)$$

A comprehensible explanation of the architecture, which the following is based on, was written in a blog post by Jay Alammar (2018). At a high structural level, the transformer consists of a stack of *n* encoders and a stack of *n* decoders. The encoders all share the same two sub-layers: a self-attention layer and a feed-forward neural network layer, to which the self-attention layers' outputs are fed. The decoders also have a third layer of attention between those two. A cardinal component of transformers, in addition to self-attention, is positional encoding. Positional encoding allows the model to account for the order of words in the input sequence. It works by adding to each input embedding an additional vector, which helps keep track of the distance between different words in the sequence and provides value to dot-product attention (Alammar, 2018 / ibid.).

One of the types of transformers are causal language models, which a trained in a supervised setting by feeding them text as input to guess the next word, e.g. GPT models Radford, et al. (2018); (2019) and (Brown, et al., 2020).

Are attention and memory in a neural network enough to enable rule-based reasoning? In 2020, the OpenAI team (Brown, et al., 2020) demonstrated with GPT-3 that transformer-based large language models are capable of learning to perform new tasks with surprisingly few examples. They concluded that extensive task-specific data or architecture is not necessary, and simply scaling up the model size and using a few-shot approach is sufficient. Among other things, they showed that language models can learn 3-digit arithmetic, a task that requires on-the-fly reasoning, just from data crawled from websites (2020 / ibid.).

Nogueira, Jiang, and Lin (2021) further studied the capabilities of transformers on arithmetic tasks. They found that the models are capable of learning addition and subtraction from very few examples if the training data is properly encoded, for example using 10e notation. However, the researchers also found that the models are unable to extrapolate the rules beyond the training parameters, except on very large models, which is unlikely to hold for more complex tasks (2021 / ibid.).

1.3 Computer Chess

As a historical pass-time and game of intellectual prowess, chess has also been one of the earliest important AI testing grounds, dating back to the works of Babbage and Turing. In 1997, IBM's Deep Blue (Campbell, Hoane, & Hsu, 2002) became the first chess machine to defeat a reigning world champion, Garry Kasparov, in a six-game match. It achieved this feat by employing a hybrid hardware/software tree-search algorithm that combined alpha-beta pruning with massive parallelism and a constant-time hardware evaluation function (Campbell, Hoane, & Hsu, 2002 / ibid.). The advancements that followed Deep Blue mostly focused on improving search algorithms. For example, David-Tabibi and Netanyahu (2008) proposed the use of null-move pruning and extended null-move reductions, which leveraged null moves (skipping a turn) as a heuristic to improve the lower-bound value for pruning in tree-search.

While all of its methods had been used before with some success, DeepMind's AlphaZero (Silver, et al., 2017) was the first to achieve state-of-the-art performance using deep neural networks with general *tabula rasa* reinforcement learning (RL), meaning that it learned everything except for the rules from playing against itself while trying to maximize positive outcomes. To work with different games, for example shogi, chess, and go, AlphaZero dropped the common alpha-beta search for a general-purpose Monte-Carlo tree search (MCTS) algorithm (Silver, et al., 2017 / ibid.), which is a probabilistic algorithm that views the game tree more broadly than alpha-beta search and estimates state values based on random simulations (Coulum, 2006).

Yet AlphaZero was not truly *tabula rasa*, since it still needed the ruleset of the underlying game. DeepMind's successor to Alphazero, MuZero (Schrittwieser J. , et al., 2020) used model-based RL, which first uses a neural network to learn a dynamic model of the underlying environment and then plans with respect to that model, to make it truly independent of prior domain knowledge while matching AlphaZero in strength.

1.4 Related Works

This work is based on a parallel approach using fine-tuned natural language processing transformer models, which have gained attention in the field in the recent years. The first chess transformer model of this kind was built in 2020 (Noever, Ciolino, & Kalin, 2020).

Noever, Ciolino, and Kalin (2020) trained a GPT-2 model on 2.8 million chess games played by players with an ELO rating over 2000. They used a stripped-down implementation of the portable game notation (PGN) format, keeping only the gametext and the Result tag, which they found useful for training. The model showed promise, as it generated games with an average of 67 moves per game, but made illegal moves 10% of the time (Noever, Ciolino, & Kalin, 2020 / ibid.).

Similarly, Stöckl (2021) analysed how many correct moves a GPT-2 model trained on just the gametext part of the PGN would make. He also showed, through neuronal analysis, that all previous moves influence the next generated move to some extent. In the discussion part of his paper, he hypothesised that the model could benefit from a longer string representation of the games (Stöckl, 2021 / ibid.). This hypothesis was tested in this thesis, along with a replication of his work, which is explained further in the methodology section.

1.5 Chess Notation

The standard plain-text computer-readable format for recording games in chess is the portable game notation (PGN), which was devised by Steven J. Edwards (1994). It consists of a minimum of seven tag pairs: "Event", "Site", "Date", "Round", "White", "Black", and "Result", followed by the movetext in standard algebraic notation (SAN). See Appendix I for an example.

In this work, the games are represented in a modified version of the Forsyth-Edwards notation (FEN), which itself is an extended version of the Forsyth standard from the 19th century (Edwards, 1994). It consists of six fields: the piece placement data, the active colour, castling availability, the en passant target square, halfmove clock, and fullmove number. In the

context of this thesis, the halfmove clock and fullmove number have been removed. See Appendix I for an example. FEN is not originally intended to represent entire games, but rather specific positions with the information necessary to restart the game from them.

The author of this work speculates that games consisting of tied-together FEN positions could better provide the information needed for language models to understand chess. Instead of just showing the model a sequence of moves, Forsyth-Edwards positions present the model with the entire board and additional information about the active colour, castling availability, and possible en passant targets. This allows the model to look at the games being played first-hand, instead of being told from a distance what moves have been made.

The trade-off of this method lies in the longer text representation of FEN compared to the SAN gametext in PGN. The longer FEN format is harder for the model to process due to the limiting nature of its context length and may cause some informational loss.

2 Data and Preprocessing

The models require a large amount of game data with legal moves, which were sourced from the open Lichess database[1]. The database holds compressed files of standard rated games played on *lichess.com* grouped by month. For this research, the data was sourced from the file for January 2023, which consists of over one hundred million played games. The dataset was decompressed using Meta's *Zstandard algorithm*[2] and sorted further with the command-line programme *pgn-extract*[3].

To control the quality of the games, the ELO values of the players in the metadata can be used to filter them. In online chess, many games are abandoned early on and therefore do not provide meaningful data for the model. A minimum length filter can be applied to remove these games from the dataset. After sorting with pgn-extract, games with less than 20 moves and an ELO rating under 2000 were excluded.

For the PGN replication dataset, all move numbers, variations, comments, results, tags, etc., were removed from the games, leaving only a pure string of SAN moves. Each game was written to the file on a single line.

The FEN dataset was further processed using a forked version of the python project *pgn-ToFen*[4], which had to be reworked to output the standard ordering of the fields and to handle all exceptions without crashing. *pgnToFen* also removed all metadata, keeping only a pure string of games states in FEN. Since FEN game states are quite long, the dataset could not be handled with standard methods when games were written to one line per game. Therefore, the games were split into batches of twelve moves with an overlap of three moves to provide shorter inputs while avoiding data loss between the sequences[5].

In both cases there was an additional small dataset, which was held out during training and never shown to the model before the testing phase. All the lines of games in the held out dataset begin from the starting position.

[1] https://database.lichess.org/
[2] https://github.com/facebook/zstd
[3] https://www.cs.kent.ac.uk/people/staff/djb/pgn-extract/
[4] https://github.com/0xsinex/pgnToFen/tree/main
[5] A video showcasing *pgnToFen* conversion: https://owncloud.ut.ee/owncloud/s/N2s92P2bBKAHXqj

3 Training and Evaluation Methods

3.1 Hardware and Use of AI-Systems

All of the computation was performed in the University of Tartu High Performance Computing Center cluster (Tartu, 2018) using NVIDIA Tesla a100 GPUs, each with 32 GB of video RAM and 512 GB of main memory.

ChatGPT (OpenAI, 2023) was used to improve text comprehension and readability with the prompt "Are there any mistakes here: [paragraph or subchapter]" and "Would you change anything to improve readability: [paragraph or subchapter]". It was also used to debug error messages with the prompt "Please explain the following error: [error message]". The model was not used to generate text, references, or code.

Grammarly (Grammarly, 2023) was used as a typing assistant to fix grammatical and typing errors while writing the thesis.

3.2 Training Methods

The models were trained using the *Pytorch* (Paszke, et al., 2019) implementation of the *Datasets* package (Lhoest, et al., 2021) and the *Transformers* package (Wolf, et al., 2020) from *HuggingFace*[6], specifically utilizing the GPT-2 architecture (Radford, et al., 2019).

Due to the long nature of the data samples and large dataset size, many of the arguments - batch sizes, gradient accumulation, and FP16 mixed-precision training (Micikevicius, et al., 2018) – were chosen to just fit on the GPU and maximize training efficiency according to *HuggingFace* efficient training recommendations[7]. The number of workers for the Dataloader were set to 0 to fix an error where the training does not start[8]. The models were saved after some steps for evaluation.

3.3 Evaluation Methods

Randomness was added to the generation process to avoid repetitive sequences that can emerge when the language model always chooses the token with the highest probability. This was done with *top-p sampling*, also known as *Nucleus Sampling*, which selects a subset of tokens that have a total probability of *p* or greater, and then samples from this subset to generate the next token (Holtzman, Buys, Du, Forbes, & Choi, 2020).

Firstly, the models are evaluated on their perplexity (Brown, et al., 1988), a statistical measure which shows how well the model predicts a given set of data. Brown et al. formulated the measure as the inverse probability of the test set, normalized by the number of words in the test set, which means that the lower the perplexity, the better the model's ability to predict data. In *Transformers* (Wolf, et al., 2020), it is calculated as the exponentiated average

[6] https://huggingface.co/
[7] https://huggingface.co/docs/transformers/main/en/perf_train_gpu_one
[8] https://github.com/pytorch/pytorch/issues/15808

negative log-likelihood of a sequence, where the sequence is considered the *i*-th token conditioned on its preceding tokens[9].

The other evaluation metrics were the same as in the work of Stöckl (2021) in order to replicate his findings and compare results. The researcher proposed using chess-specific evaluation methods, calculating the average count of correct moves on games generated in three different ways:

- From a list of typical opening positions after two moves;
- From positions of games from a game data set after a given number of moves;
- From randomly-generated positions after a given number of moves.

According to the author, the three chess-specific metrics test different aspects of the model's generalization capabilities. For the first metric, he says, that since all of the opening positions are found in the training data, the model can choose moves from its memory and won't need any inherent knowledge of the rules until the sequences become very long.

In the second evaluation method, the model will increasingly encounter more positions that it hasn't seen before, since the original positions are sampled from a game dataset held out during training. Stöckl hypothesises that performance on this metric should scale with understanding of the ruleset.

The sequence of random moves generated for the third metric will be the most challenging metric for the models because most of these moves have never appeared in human games, nor in the training data. The move patterns are very different from regular games and therefore, the author says, it is very difficult for the model to generate legal moves for these sequences (Stöckl, 2021 / ibid.). A widely popular psychology study by Chase and Simon (1973) showed that amateur chess players do not perform better at remembering random positions than masters, since they cannot rely on the relational structures between the pieces that they use ordinarily. Language models with only attention and memory, therefore, will also not be able to directly rely on small-scale representations of sequences and would need an abstract understanding of the game to make legal moves from random positions.

[9] https://huggingface.co/docs/transformers/perplexity

4 Results

A total of four GPT-2 models were trained – the differing factors being format and dataset size. The FEN datasets consisted of 1,111,801 games for the larger model and 555,900 games for the smaller model. The PGN datasets consisted of 1,100,000 games and 550,000 games respectively. To assess the results, the models were saved after every 500 training steps and evaluated according to the methods described in chapter 4.3. A link to the code repository can be found in Appendix II. For all of the combinations of models and evaluation methods, the results were plotted over the training steps with a logarithmic regression fit drawn for each model.

4.1 Forsyth-Edwards Notation Models

The perplexity scores of the models, which can be seen on Figure 1., show that the FEN models outperformed the PGN models. Still, showing an upwards trend with increasing data and training indicates that performance of the FEN models would decrease and likely collide with the PGN models further on. While a perplexity score of *k* means that the model is as confused as if it had to choose uniformly among *k* possibilities (Brown, et al., 1988), this may not necessarily be a negative aspect for a chess model. It can imply that the models are not learning the rules and are simply memorizing potential moves for the sequence. In the context of chess though, it may also mean that the models are capable of generalizing better and suggest more potential moves, many of which could be legal.

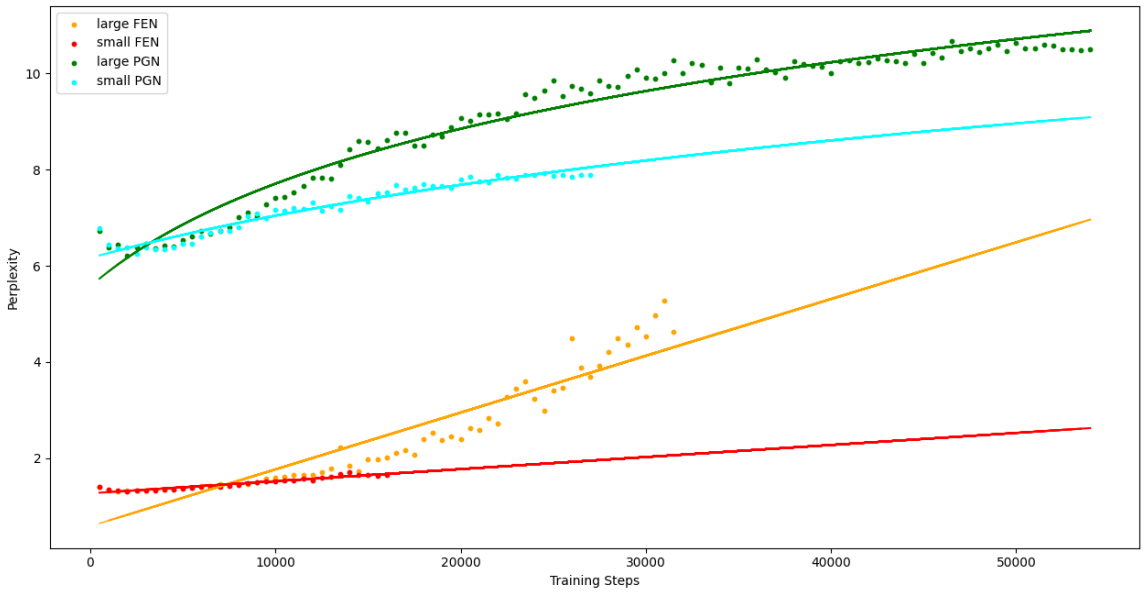

Figure 1. Perplexity scores over the training period

Clear and reoccurring patterns emerge when the models are tested on each of the chess-specific metrics, see the graphs on Figures 2, 3, and 4. Models trained on FEN data would consistently make at least one correct move only from typical opening positions, which had all certainly appeared in the training data. PGN models, on the other hand, fared much better and consistently made less than five correct moves only in positions that resulted from ten random moves.

Using FEN data does not provide an advantage to the models over using the shorter PGN and, in fact, inhibits the model from generating legal moves consistently. Moreover, the FEN models do not show improvement after the initial training steps nor do they show any remarkable differences in accuracy in respect to dataset size.

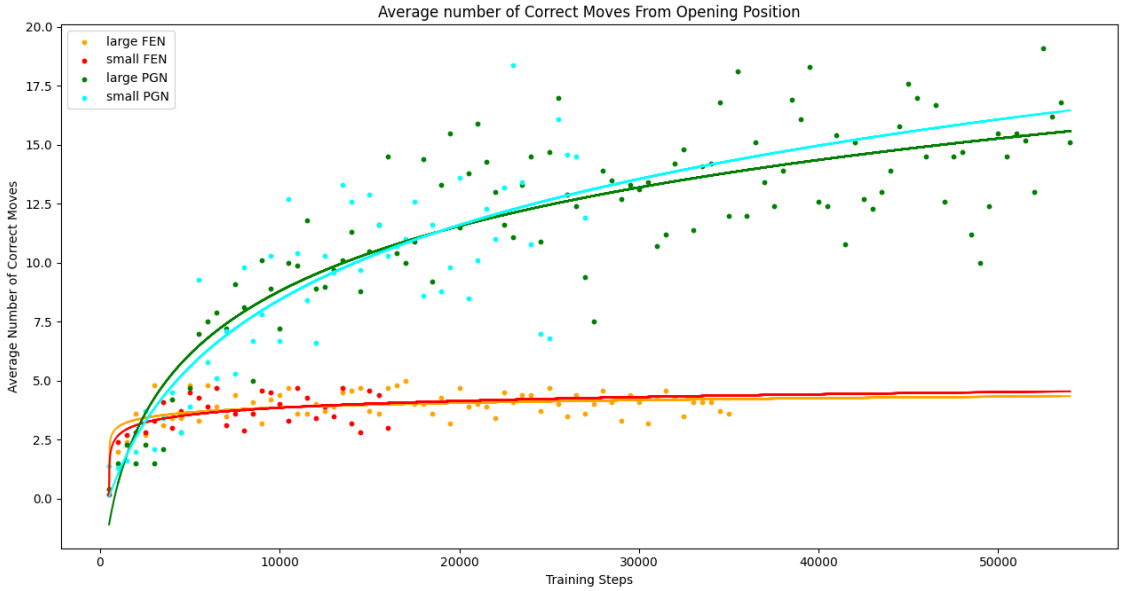

Figure 2. Average number of correct moves generated from typical opening positions

Figure 3. Average number of correct moves generated from positions after 10 random moves

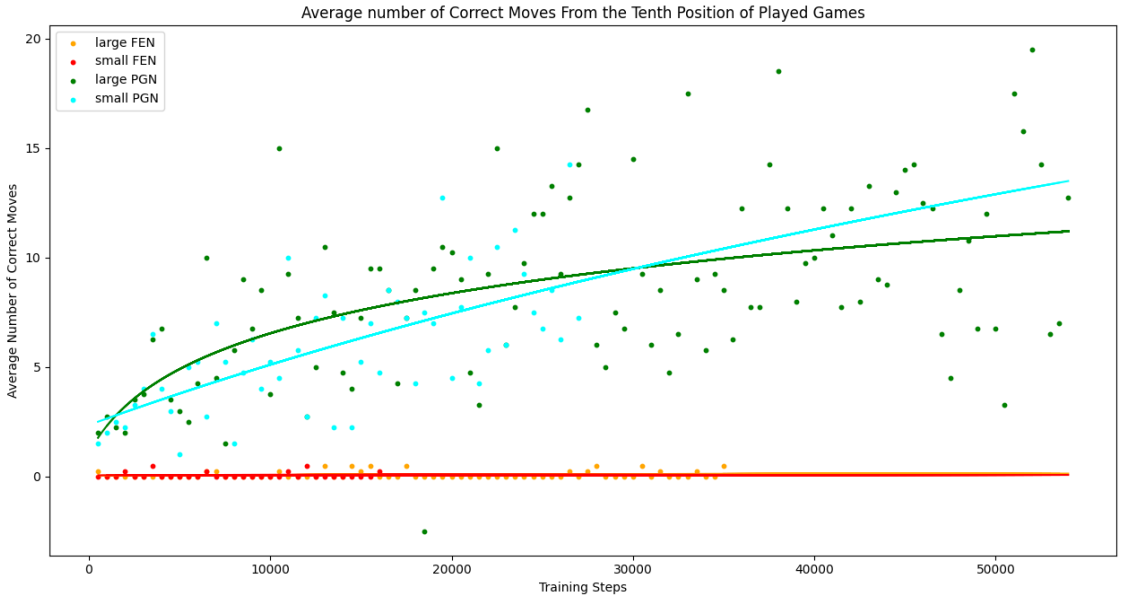

Figure 4. Average number of correct moves generated from the tenth position of played games

The generated moves of the FEN models are often very similar to legal moves, but the mistakes indicate a misunderstanding of the basic underlying rules. For example, the models might generate a position with a valid move, but mark a line on the board as having more than eight squares or mark more pieces than there are in the game. Further analysis is needed to identify potentially emerging patterns of mistakes, which might be worked around.

The models are not capable of understanding the whole chessboard and its underlying ruleset from FEN datasets that are comparable in size to PGN datasets. While longer representations of games might be beneficial, the complexity of the whole chessboard for each position, rather than simply a move, is too much for a relatively small language model to comprehend.

Another aspect that could lead to improvement is the use of larger datasets. FEN data is more complex than PGN as one move consists of around 50 characters, compared to two to seven characters in PGN, and provides much more information about the state of the game. It is reasonable to assume that an exponentially larger dataset could help address the challenges in learning, but it would increase computational complexity beyond what can be deemed rational when there are alternative solutions to the problem.

4.2 Replication

The medium GPT-2 models trained on 577,202 games and 2,163,417 games by Stöckl (2021) are the most comparable to the PGN models, trained with around half of a million and one million games respectively, in this thesis. In his work, training hours were used instead of steps for the basis of when to evaluate models. In this thesis, the models finished training at 14 hours for the smaller and 22 hours for the larger model. The comparisons, which can be seen in Table 1, were made at the logarithmic regression value at 20 hours of

training for Stöckl's models and at the logarithmic regression value at the final training step of the replicated models.

Table 1. Comparison of the Replication Models

	577,202 games (Stöckl, 2021)	2,163,417 games (Stöckl, 2021)	550,000 games	1,100,000 games
From Opening Positions	28	20	12.5	13
Position After 10 Random Moves	2.8	2.9	2.7	3.25
Position 10 of Played Game	10.4	11	8	11

While the comparisons are fairly arbitrary, it can be seen that the replication models achieved similar results in generating legal moves after the 10^{th} positions of games, yet were significantly outperformed when generating moves from typical opening positions. Since the models were trained on identically formatted datasets and evaluated using the same code, the differences may arise from the use of different hyper-parameters during the training process.

Summary

To test whether a longer text representation of chess moves would improve a language model's understanding of chess rules, two types of GPT-2-based transformer models, two for FEN data and two for PGN data, with comparable numbers of games were trained. The models were tested and compared on perplexity and chess-specific metrics. The previously suggested PGN models (Stöckl, 2021) outperformed the models proposed in this work by a clear margin in each chess-specific category.

The author concludes that language models are capable of learning to generate correct chess moves to a certain extent from seeing unannotated transcripts of games. While this thesis explored using a richer text format, which includes a representation of the whole chessboard for each move, shorter formats remain a better option for natural language processing models.

Further research could explore the internal mechanisms of the models, to see which parts of the Forsyth-Edwards notation are most important to the generation of correct moves. In other words, it remains to be answered, which parts of the input sequence the transformer pays attention to during move generation and how the representations are stored in the memory.

References

Alammar, J. (27. June 2018. a.). The Illustrated Transformer. Allikas: https://jalammar.github.io/illustrated-transformer/

Bhatt, S. (19. March 2018. a.). Reinforcement Learning 101. *Towards Data Science*. Allikas: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Brown, P., Cocke, J., Pietra, S., Pietra, V., Jelinek, F., Mercer, R., & Roossin, P. (1988). A Statistical Approach to Language Translation. *International Conference on Computational Linguistics*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., & al., e. (2020). Language Models are Few-Shot Learners. *arXiv*. doi:https://doi.org/10.48550/arXiv.2005.14165

Campbell, M., Hoane, A. J., & Hsu, F.-H. (2002). Deep Blue. *Artificial Intelligence*, 57-83. doi:https://doi.org/10.1016/S0004-3702(01)00129-1

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology, 4*(1), 55-81. doi:https://doi.org/10.1016/0010-0285(73)90004-2

Coulum, R. (2006). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Computers and Games*, 72-83. doi:10.1007/978-3-540-75538-8_7

David-Tabibi, O., & Netanyahu, N. (2008). Extended Null-Move Reductions. *Computers and Games*, 205-216. doi:10.1007/978-3-540-87608-3_19

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., & Sutskever, I. (2020). Jukebox: A Generative Model for Music. *arXiv*. doi:https://doi.org/10.48550/arXiv.2005.00341

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Xiaohua, Z., Unterthiner, T., . . . Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv*. doi:https://doi.org/10.48550/arXiv.2010.11929

Edwards, S. J. (12. March 1994. a.). Portable Game Notation Specification and Implementation Guide. Allikas: https://ia802908.us.archive.org/26/items/pgn-standard-1994-03-12/PGN_standard_1994-03-12.txt

Grammarly. (2023). Grammarly [AI typing assistant]. Allikas: https://www.grammarly.com/

Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The Curious Case of Neural Text Degeneration. *arXiv*. doi:https://doi.org/10.48550/arXiv.1904.09751

IBM. (2023). *What is supervised learning?* (IBM) Allikas: IBM: https://www.ibm.com/topics/supervised-learning

IBM. (2023). *What is Unsupervised Learning?* Allikas: IBM: https://www.ibm.com/topics/unsupervised-learning

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., & al., e. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature, 596*(7873), 583-589. doi:doi:10.1038/s41586-021-03819-2

Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., . . . Wolf, T. (2021). Datasets: A Community Library for Natural Language Processing. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 175-184. doi:10.18653/v1/2021.emnlp-demo.21"

Martin, K. (2012, November 01). *Common Mistakes Made When Writing a Book in Microsoft Word.* Retrieved from Self-pub by Jera Publishing: http://www.self-pub.net/blog/common-mistakes-made-when-writing-a-book-in-microsoft-word/

Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., . . . Morimoto, J. (2022). Deep learning, reinforcment learning, and world models. *Neural Networks, 152*, 267-275. doi:https://doi.org/10.1016/j.neunet.2022.03.037.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115-133. doi:10.1007/BF02478259

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., . . . Wu, H. (2018). Mixed Precision Training. *arXiv*. doi:https://doi.org/10.48550/arXiv.1710.03740

Noever, D., Ciolino, M., & Kalin, J. (2020). The Chess Transformer: Mastering Play using Generative Language Models. *arXiv*. doi:https://doi.org/10.48550/arXiv.2008.04057

Nogueira, R., Jiang, Z., & Lin, J. (2021). Investigating the Limitations of Transformers with Simple Arithmetic Tasks. *arXiv*. doi:https://doi.org/10.48550/arXiv.2102.13019

OpenAI. (2023). ChatGPT [Large Language Model]. Allikas: https://chat.openai.com/chat

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., . . . Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv*. doi:https://doi.org/10.48550/arXiv.1912.01703

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. Allikas: https://gwern.net/doc/www/s3-us-west-2.amazonaws.com/d73fdc5ffa8627bce44dcda2fc012da638ffb158.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. Allikas: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review, 65*(6), 386-408. doi:10.1037/h0042519

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., . . . Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature, 588*, 604-609. doi:https://doi.org/10.1038/s41586-020-03051-4

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Lanctot, M., . . . Hassabis, D. (2017). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv*. doi:10.48550/arXiv.1712.01815

Sinai, J. (11. November 2017. a.). The Perceptron. Allikas: https://jontysinai.github.io/jekyll/update/2017/11/11/the-perceptron.html

Stöckl, A. (2021). Watching a Language Model Learning Chess. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, 1369–1379. Allikas: https://aclanthology.org/2021.ranlp-1.153

Tartu, U. o. (2018). UT Rocket. doi:https://doi.org/10.23673/PH6N-0144

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *arXiv*. doi:https://doi.org/10.48550/arXiv.1706.03762

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38-45. doi:10.18653/v1/2020.emnlp-demos.6

Appendix

I. Examples of Formats

The game used for both examples is Efim Keller vs Paul Keres in the USSR Championships in 1951, where Keres successfully defended his USSR Champion title[10].

The game in PGN:

[Event "USSR Championship"]
[Site "Moscow URS"]
[Date "1951.11.18"]
[EventDate "1951.??.??"]
[Round "4"]
[Result "0-1"]
[White "Efim Geller"]
[Black "Paul Keres"]
[ECO "C99"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "62"]

1.e4 e5 2.Nf3 Nc6 3.Bb5 a6 4.Ba4 Nf6 5.O-O Be7 6.Re1 b5 7.Bb3
d6 8.c3 O-O 9.h3 Na5 10.Bc2 c5 11.d4 Qc7 12.Nbd2 cxd4 13.cxd4
Bb7 14.Nf1 Rac8 15.Bb1 d5 16.exd5 exd4 17.Bg5 h6 18.Bh4 Nxd5
19.Qd3 g6 20.Bg3 Bd6 21.Bxd6 Qxd6 22.Qd2 Nf4 23.Qxa5 Bxf3
24.gxf3 Nxh3+ 25.Kg2 Nf4+ 26.Kg1 Nh3+ 27.Kg2 Nf4+ 28.Kg1 Qd5
29.Ng3 d3 30.Ne4 Qf5 31.Qb4 Rfe8 0-1

The first eight moves of same game in the adapted FEN format used for training:

rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -;
rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3;
rnbqkbnr/pppp1ppp/8/4p3/4P3/8/PPPP1PPP/RNBQKBNR w KQkq e6;
rnbqkbnr/pppp1ppp/8/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq -;
r1bqkbnr/pppp1ppp/2n5/4p3/4P3/5N2/PPPP1PPP/RNBQKB1R w KQkq -;
r1bqkbnr/pppp1ppp/2n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R b KQkq -;
r1bqkbnr/1ppp1ppp/p1n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R w KQkq -;
r1bqkbnr/1ppp1ppp/p1n5/4p3/B3P3/5N2/PPPP1PPP/RNBQK2R b KQkq -;

[10] https://www.chessgames.com/perl/chessgame?gid=1072805

II. GitHub Repository of the Used Code

https://github.com/RMarrandi/Chess-GPT2

III. License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Raiko Marrandi,

(author's name)

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

Evaluating Transformer Architecture for the Game of Chess,

(title of thesis)

supervised by PhD Eduard Barbu.

(supervisor's name)

2. I grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace

digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in points 1 and 2.

4. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Raiko Marrandi

09/05/2023