Mastering Chess Through Natural Language Models

Helder Vieira

November 2021

Abstract

Natural language tasks in Machine Learning are dominated by models that implement the transformer architecture, such as BERT, T5 and GPT-2. There is no doubt that this approach fits excellently well to perform tasks such as translation, text generation, sentiment analysis, among others traditional cases. The goal of this experiment is to measure, in a qualitative way, the capability of these models to generalize complex rules and environments. To achieve this, we used a well-known deterministic closed domain: the chess game. We trained with 1 epoch a GPT-2 model with 99M parameters in a dataset with about 2.2M chess games in the PGN representation, where the game is represented by a sequence of moves (tokens) and read from left to right. The study suggested that the model was capable of performing a reasonable generalization, playing well known opening positions or usual movements, but struggling with basic out-of-the-standard positions, thus being incapable of beating a human player.

1 Introduction

Transformers are dominating the great field of Machine Learning since the past few years. One had the first big appearance in *Attention Is All You Need* [1], when its use guaranteed the start of some breakthroughs in Natural Language Processing tasks. More recently, people started to use the same architecture in other fields, such as image processing and speech recognition. This type of model proved to be the way-to-go when building solutions with non structured data, such as texts, audios and images, getting outperforming results when compared to other classical techniques such as LSTMs and RNNs. The models are usually pre-trained in a huge amount of data, and then fine tuned in a specific task e.g text classification or language translation. They used to have from tens of millions to billions of parameters - GPT-2 [2], a language model, in its biggest version has 1.5B parameters that need a high amount of data to be tuned. As these kinds of models are trained in such a high variety of examples, one question that arises is whether the model is actually learning the world or is it just memorizing everything.

In this experiment, we suggest a way to measure the capabilities of generalization of a GPT-2 model pre-training it from scratch in a deterministic and closed domain: the chess game. We are going to use this model to

generate moves and play games against it, understanding the limitations of its learnings. A similar work was published recently [3], with convergence in the way that the strings were used, but with significantly different approaches in the tokenization step.

2 Methodology

To be able to use a language model such as GPT-2 to play chess it is important to represent the game with text. There is a representation named Portable Game Notation - PGN, which is the way that professional players annotate their moves in a game. One example of a PGN is shown in Figure 1. It is split in two main parts: on the top, a header, containing information about the game, like the result, the players names etc. On the bottom is the representation of the game, where each turn is written as <turn_number>. <white_move> <black_move>.

```
[Site "chess24.com"]
[Date "2015.02.02"]
[Round "1"]
[White "Caruana, Fabiano"]
[Black "Anand, Viswanathan"]
[Result "1/2-1/2"]
[WhiteElo "2820"]
[BlackElo "2797"]
[PlyCount "76"]
[EventDate "2015.??.??"]

1. e4 e5 2. Nf3 Nc6 3. Bc4 Bc5 4. c3 Nf6 5. d3 d6 6. b4 Bb6 7. a4 a5 8. b5 Ne7
9. O-O O-O 10. h3 c6 11. Bb3 Ng6 12. Re1 Re8 13. Nbd2 d5 14. Nf1 dxe4 15. Ng5
Re7 16. Nxe4 Nxe4 17. dxe4 Be6 18. Rb1 Rd7 19. Qc2 Nf8 20. Ne3 Bxe3 21. Bxe3 h6
22. Red1 Rc8 23. Rxd7 Qxd7 24. Rd1 Qe8 25. bxc6 Qxc6 26. Bd5 Qxc3 27. Qxc3 Rxc3
28. Bxb7 Nd7 29. Ba6 Nc5 30. Bxc5 Rxc5 31. Bb5 Kh7 32. Rd6 Bc4 33. Rb6 Bxb5 34.
Rxb5 Rxb5 35. axb5 a4 36. b6 a3 37. b7 a2 38. b8=Q a1=Q+ 1/2-1/2
```

Figure 1: Example of PGN. This game ended in a draw

The only part needed to perform the experiment is the second one. Besides that, not all of it is necessary, only the moves. The number indicating the turn is not used because we assume that this information is already captured by the model with the positional embeddings. This leads to a representation that is just as simple as a sequence of moves for each game.

As the game of chess is a closed domain, there is a finite number of valid moves, so it is possible to perform a word tokenization with a vocabulary that covers all tokens, making the use of unknown tokens unnecessary. Each move is composed of some parts, which may be described as: <piece><disambiguity_indicator><capture_indicator><final_square><promotion><check_or_mate_indicator>.

The possible characters for each part is as follows:

- piece: Rook (**R**), Knight (**N**), Bishop (**B**), Queen (**Q**), King (**K**), Pawn (**empty**);
- disambiguity_indicator: column (from **a** to **h**) or row (from **1** to **8**) or (**empty**);
- capture_indicator: (**x**) if capture, (**empty**) if not;
- final_square: column (from **a** to **h**) plus row (from **1** to **8**);
- promotion: (**=<piece>**);
- check_or_mate_indicator: check (**+**), mate (**#**).

With this and hands, all possible combinations can be made in order to build the chess vocabulary that, with the addition of the special tokens ([PAD], [UNK], [MASK], [CLS] e [SEP]), has **17675** different tokens.

An example of a game sequence, driven from the PGN file, can be found as this:

d4 Nf6 Nf3 g6 Bf4 Bg7 e3 O-O Bd3 d6 O-O Nbd7 c3 Re8 Qe2 e5 dxe5 dxe5 Bg3 e4 Bxe4 Nxe4

Which can be split with simple white space and then tokenized with the chess vocabulary.

The goal of this study is to train the GPT-2 model in order to predict the next move given the sequence of the past n moves, tokenized with the chess vocabulary. In other words, to get the **next legal** move with the highest probability among all next legal moves. In this way, it is possible to play games against the model.

3 Data set

The dataset used for the experiment was obtained by the collection of games published by The Week In Chess portal [2]. Every week, there are many games - most of them played by professional players - posted on the website. A 9GB collection of games was consolidated in a single text file. A slice of the file, approximately 3.5GB was indeed used for the experiment, for time

performing reasons. One of the items of the future work is to perform the training with more data, and the use of the whole dataset is a good option to start.

The data could not be used as it is, because it contains the raw PGN, so an ETL step was performed. This step was divided in two parts: first, an extraction of the PGNs from the raw dataset, consolidating them in small files of 6000 games each. Indeed, while the PGN was saved in a .csv file with the string sequence and an ID, the meta information of the game was saved in a different file, carrying the results and other useful information. This was done in order to make it easier (due to the file size) to get statistics about the dataset. The second part was to filter the PGN string of each game, which may include some undesirable characters, such as text commentaries, and unnecessary tokens, such as the turn indicator. Games that eventually could not be filtered through simple regex expressions, were discarded with a simple check if there was any token that was not present in the chess vocabulary. The result of this processing was then saved in two files: one for train with 2.2M examples, other for validation with 200K examples. Each game was separated by a line in the final dataset.

4 Experiments

All experiments were done in Python using the transformers library from HuggingFace. The GPT-2 hyperparameters were used as the default, resulting in almost **100M** parameters, a number quite equivalent to the small version of the model. There were two versions of the model. The first one, the inputs were just the string representation as it is, called version without pre-movements. The second, called version with pre-movements, was put on a prefix string in order to represent the piece's initial positions. The training step was performed with only 1 epoch in both versions and took about 14 hours to complete in the version without pre-movements, reaching a training loss of **2.43,** and 16 hours in the version pre-movements, reaching a training loss of **1.7**. Both versions were trained in a Tesla P100 with 16GB VRAM available, with a batch size of 16 examples and padding on 256 tokens.

To measure the performance of the model on playing chess, an environment was built using the python-chess library. For each turn, the model put probabilities in a list of valid moves and the token with the highest probability was chosen. The model struggled to win just a single game against a human with basic knowledge of chess, and the possible reasons are discussed in the next section. Nevertheless, a classic approach was used to understand how well the trained model was on playing the game: by evaluating partial games.

That was made using the analysis board on the Chess.com website, which uses some heuristics to evaluate the position. The evaluation metric is a float number that is positive if the advantage is with the player playing with the white pieces, and negative if it is with the player with the black ones. For positions that are forcing winning (there is a forcing sequence of checkmate), the evaluation will be **W**, when it's towards white favor, and **B** otherwise. The following table shows the evaluation for 6 games for each version. In all games, the model was playing with the black pieces. The 6th games were played by a person with little knowledge about chess - almost only the basic movement rules. It is important to mention that, for professional players, an advantage of +/- 2.0 is already considered decisive.

```
a2 a7 b2 b7 c2 c7 d2 d7 e2 e7 f2 f7 g2 g7 h2 h7 Ra1 Ra8
Nb1 Nb8 Bc1 Bc8 Qd1 Qd8 Ke1 Ke8 Bf1 Bf8 Ng1 Ng8 Rh1 Rh8
```

Figure 2: Pre-movements simulating the first positioning of the pieces on the board.

Table 1: Partial evaluations of the games of the model without pre-movements.

	Move 10	Move 20	Move 30	Move 40	Move 50
Game 1	-4.4	-5.3	8.1	W	-
Game 2	-0.1	-8.2	B	W	-
Game 3	-0.3	1.6	-1.9	2.4	W
Game 4	0.0	-6.0	-0.2	0.2	W
Game 5	-3.3	-0.2	-6.6	-3.7	0.0
Game 6	-2.1	-6.4	-10.0	W	-

Table 2: Partial evaluations of the games of the model with pre-movements.

	Move 10	Move 20	Move 30	Move 40	Move 50
Game 1	-0.5	-1.9	-6.1	-1.7	0.1
Game 2	0.0	0.9	5.7	W	-
Game 3	0.0	-0.1	10	W	-
Game 4	-2.0	-0.2	-0.9	W	-
Game 5	0.0	-8.8	-9.9	B	-
Game 6	-2.2	-8.3	3.5	0.1	-1.8

5 Conclusion

The tables 1 and 2 show that apart from generating good moves in the beginning of the match, the model could not keep it until the end, making a win. It is important to keep in mind that for each move in chess, there are about 25 options to be chosen. That simple fact leads to about $(25)^{60}$ possible games up to move 30. That is a huge number and there is a small chance that one of the 12 games played up to this turn would have appeared in the 2.3M games in the training set. The capability of the model to play a reasonable chess game up to this turn is evidence that the model can generalize the sequences learned - and also store a kind of representation of the board, in order to decide what to do. However, the game of chess allows small margins of error: a single wrong move can (and probably will) lead to a losing configuration. That is one of the reasons that the model is very unlikely to win against any human person that knows the basics of the game.

That was also clear that the model could not convert winning positions - in other words, to apply a checkmate. The reason for this is that professional games, which were the most part of the training dataset, rarely end in checkmate: when a player realizes that the position is already lost - often when the evaluation is close to +/- 2.0, there is a resignation. So one could

say that the model is actually learning to get a winning position, but not to actually win. If the human player doesn't resign, sooner or later the model makes a mistake and loses.

There was no significant difference between the results of the model that starts with the pre-movements sequence and the one that takes the string directly. However, the games were slightly different, with the first one making less mistakes such as leaving a piece to be taken for free and winning a game by taking an opportunity over a huge mistake. All game sequences are available in the appendix section.

6 Future Work

There is a lot to be done in order to study the characteristics of the trained model: understanding the way that the model makes mistakes could lead to a better knowledge of how language models work. An automatic technique would be necessary to generate more scenarios, so a statistically significant analysis could be made.

In order to tell if the reason that the model could not apply a checkmate, it would be great to train a new model with a different dataset, filtered to contain only games that were played until the last move. If the hypothesis is correct, this would probably lead to a better generalization and perhaps a better-than-regular-human performance. The challenge is to gather more data - or generate them by using open source chess engines, as the percentage of high level games played until the end in real life is small.

To achieve a better performance, two possible approaches could be taken: a fine-tuning step, using the evaluation of the position as a regression target. The other approach, harder for course, would be to implement a reinforcement learning algorithm to train the model. In that way, the model could learn to generate sequences that beat itself. This last approach is taken in the best chess algorithms, such as the AlphaZero [4], but they use conventional CNN's.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (ed.), Advances in Neural Information Processing Systems 30 (pp. 5998–6008) . Curran Associates, Inc. .

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (ed.), Advances in Neural Information Processing Systems 30 (pp. 5998–6008) . Curran Associates, Inc. .

[4] Stöckl, A. (2021, September). Watching a Language Model Learning Chess. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021) (pp. 1369-1379).

Appendix

Dataset

https://zenodo.org/record/5767528#.YbFXYPHMKi4

Google-Colab notebooks

In order to execute the notebooks to reproduce the experimental results, the file locations need to be changed.

https://colab.research.google.com/drive/1NMb08ov8VeU6YlioRanTas43nFr1RJrK?usp=sharing

https://colab.research.google.com/drive/1Zr-JGft8tUrYEk5p96nKpni04-84DJXS?usp=sharing

https://colab.research.google.com/drive/1znGBJZbSOWptkKaA4FgNsaaxNfzxt_I9?usp=sharing

Github project

https://github.com/heldervj/nlp-chess

Games Model Without Pre-Movements

Game 1

d4 Nf6 Nf3 g6 Bf4 Bg7 e3 O-O Bd3 d6 O-O Nbd7 c3 Re8 Qe2 e5 dxe5 dxe5 Bg3 e4 Bxe4 Nxe4 Bf4 Qe7 Nbd2 Ndf6 Nxe4 Nxe4 h3 Bf5 g4 Bd7 Rad1 Bc6 Nd2 Nxd2 Rxd2 Rad8 Rfd1 Rxd2 Qxd2 Rd8 Qxd8 Qxd8 Rxd8 Bf8 Bxc7 Kg7 Bd6 f5 Bxf8 Kf6 gxf5 gxf5 Bc5 a6 Rf8 Ke6 Rh8 Kd5 Bd4 Ke4 Rxh7 Kd3 Rh5 Kc2 Rxf5 Kxb2 h4 Kxa2 h5 Kb3 h6 a5 h7 a4 h8=Q a3 Ra5 b5 Rxa3 Kc4 Qg8+ Kd3 Qc8 Ke4 Qxc6+ Kd3 Qxb5+ Ke4 c4 Kf3 Qf5+ Ke2 Ra2+ Ke1 Qb1

Game 2

e4 c5 Nf3 d6 d4 cxd4 Nxd4 Nf6 Bd3 Nc6 Nxc6 bxc6 O-O e5 Bg5 Be7 Nc3

O-O Bc4 Be6 Bxe6 fxe6 Re1 Qc7 h3 Rad8 b4 d5 exd5 cxd5 Rb1 d4 Ne4 Nxe4 Rxe4 Bxg5 h4 Bh6 g4 Bf4 f3 Qf7 h5 h6 Qe1 Bg5 Qf2 Bf4 Rb3 Rc8 Qe1 Rxc2 b5 d3 Qd1 d2 b6 axb6 Rxb6 Rc1 Qxc1 d1=Q+ Qxd1 Be3+ Rxe3 Qxf3 Rxf3 Rxf3 a4 Ra3 Rxe6 Rxa4 Qd8+ Kh7 Qe8 Rf4 Qg6+ Kh8 Re8+ Rf8 Rxf8#

Game 3

g4 d5 Bg2 e5 f3 Bc5 h4 h5 d3 hxg4 fxg4 Bxg4 Bf3 Bxf3 Nxf3 e4 dxe4 dxe4 Qxd8+ Kxd8 Ng5 Nh6 Nxe4 Bb6 c4 Nf5 Ng5 Nc6 Nxf7+ Ke7 Nxh8 Rxh8 Bg5+ Ke6 Nc3 Ne5 Kd2 Nxc4+ Kc2 Nd4+ Kd1 Ne3+ Kc1 Nc4 Be3 Nxe3 Nd1 Nxd1 Rxd1 Nf5 b3 Ne3 Rd3 Ng4 Kb2 Ne5 Rc3 c6 Rg3 Kf5 Rxg7 Rxh4 Rxb7 Rh2 Rf1+ Ke6 Re1 Kd6 Ka3 a5 Rb8 Kc7 Re8 a4 Rxe5 axb3 axb3 c5 b4 cxb4+ Kxb4 Kc6 Rc1+ Kb7 Kb5 Ka7 Re7+ Ka8 Rc8#

Game 4

e4 c5 Ne2 Nc6 d4 cxd4 Nxd4 Nf6 Be2 e5 Nf5 d5 O-O Bxf5 exf5 d4 Bg5 Be7 c3 O-O cxd4 exd4 Bg4 Nxg4 Qxg4 Bxg5 h4 Bf6 Nd2 Re8 Rfe1 Ne5 Qe4 Qb6 b3 Rad8 Rac1 d3 Kh2 Qd4 Qxd4 Rxd4 Re4 Rxe4 Nxe4 Bxh4 g3 Be7 Rc7 Kf8 Rxb7 a5 a3 h5 b4 axb4 axb4 Bxb4 Rxb4 d2 Nxd2 Nf3+ Nxf3 Re2 Rb8+ Ke7 Rb7+ Kf6 Kh3 Rxf2 Rb6+ Kxf5 Rb5+ Kg6 Ne5+ Kf6 Ra5 g6 Kh4 Rf5 Nc6 Kg7 Rxf5 gxf5 Kxh5 Kf6 Nd4 Ke5 Nf3+ Kf6 Nh4 Kg7 Nxf5+ Kf6 g4 Ke5 Nh6 f6 Ng8 Ke6 Nxf6 Kxf6 g5+ Kg7 g6 Kg8 Kh6 Kh8 g7+ Kg8 Kg6

Game 5

d4 Nf6 e3 g6 Nc3 Bg7 Bd2 O-O Qe2 d6 O-O-O c6 g3 b5 f4 b4 Nb1 a5 Bg2 Ba6 Nh3 Qb6 Ng5 d5 e4 dxe4 Nxe4 Nxe4 Qxe4 Nd7 Qxe7 Nf6 d5 cxd5 Be3 Qb7 Bc5 Rfe8 Qxb7 Bxb7 Rhe1 Ne4 Nd2 Nxc5 Rxe8+ Rxe8 Nf3 Re2 Rd2 Rxd2 Kxd2 Ne6 Ne5 Bxe5 fxe5 Kf8 c3 bxc3+ Kxc3 Ke7 b4 axb4+ Kxb4 Ba6 a4 Kd7 a5 Kc6 h4 h5 Bh3 d4 g4 hxg4 Bxg4 d3 h5 gxh5 Bxh5 d2 Kc3 Kd5 Kxd2 Kxe5 Bxf7 Nd4 Kc3 Nb5+ Kb4 Nc7 Bc4 Kd6 Bxa6 Nxa6+ Kb5 Nc7+ Kb6 Nd5+ Kb7 Ne7 a6 Nc6 Kb6 Nb8 a7 Nd7+ Kb7 Nc5+ Kb8 Nd7+ Kc8 Kc6 a8=Q+ Kd6

Game 6

d4 Nf6 c4 e6 Bd2 d5 cxd5 exd5 Qa4+ c6 e3 Bd6 Ba5 Qe7 Nc3 O-O O-O-O b5 Qc2 b4 Nce2 Ba6 b3 Bxe2 Bxe2 c5 dxc5 Bxc5 g3 Nc6 Nf3 Rac8 Nh4 g6

f3 Ne5 Rhe1 Rfe8 h3 Qb7 f4 Ned7 g4 Ne4 Qb1 Nc3 Qb2 Nxd1 Rxd1 Nf6 Qc2 Ne4 f5 Nc3 Rd2 Nxe2+ Kb2 Nd4 Qd3 Re4 fxg6 hxg6 Ng2 Ne6 g5 d4 exd4 Nxd4 Ne1 Qc6 h4 Nf5 Re2 Nxh4 Rxe4 Qxe4 Qxe4 Ng2 Nxg2 Re8 Qxe8+ Kg7 Qc8 Bb6 Bxb6 axb6 Qb8 b5 Qxb5 f5 Qxb4 f4 Qxf4 Kh7 Qf6 Kg8 Qxg6+ Kf8 a4 Ke7 a5 Kd7 a6 Kc7 Qh6 Kb8 Qh7 Ka8 g6 Kb8 g7 Ka7 g8=Q+ Kb6 Qb8+ Kc6 Qh6+ Kd7 Qb7+ Kd8 Qh8#

Games Model With Pre-Movements

Game 1

d4 Nf6 Bf4 g6 Nf3 Bg7 e3 O-O Bd3 d6 c3 Nbd7 O-O Re8 Na3 e5 dxe5 dxe5 Bg3 Nh5 Nd2 Nxg3 hxg3 Nc5 Bc4 Be6 Bxe6 Rxe6 b4 Nd7 Ne4 Qe7 f4 exf4 exf4 f5 Ng5 Re3 Re1 Rxe1+ Qxe1 Re8 Qxe7 Rxe7 Kf2 Nf6 Nb5 c6 Nxa7 Ne4+ Nxe4 Rxe4 Re1 Rxe1 Kxe1 Bxc3+ Kd1 Bxb4 Kc1 Ba3+ Kc2 Kf7 Kb3 Bd6 Kc4 Ke6 Kb3 Kd5 Ka4 b6 Nc8 Bc5 Kb3 b5 Kc3 b4+ Kd3 b3 axb3 Bg1 Ne7+ Kd6 Ng8 c5 Ke2 c4 bxc4 Kc5 Kf3 Kxc4 g4 fxg4+ Kxg4 Kd3 Nf6 h6 f5 gxf5+ Kxf5 Bd4 Kg6 Bxf6 Kxf6 Ke4 Kg6 Kf4 Kxh6 Kg4 Kg6 Kf4 Kh5 Kf5 g3 Kf6 g4 Kf7 g5 Kg7 g6 Kg8 Kh6 Kh8 g7+ Kg8 Kh5 Kh7 Kg5 Kg8 Kh5 Kh7 Kg5 Kg8

Game 2

e4 c5 Nf3 d6 Bb5+ Bd7 Bxd7+ Qxd7 O-O Nc6 d4 cxd4 Nxd4 Nf6 Re1 g6 Bg5 Bg7 Nxc6 bxc6 b3 O-O c4 h6 Bh4 g5 Bg3 Nh5 Re3 Nxg3 Rxg3 f5 Nd2 f4 Rh3 e5 Qh5 Qf7 Qxf7+ Rxf7 g3 fxg3 hxg3 Raf8 Rf1 Rf3 Nxf3 Rxf3 Kg2 Rd3 Rfh1 Kf7 b4 Ke6 c5 dxc5 bxc5 Bf8 Rc1 Kd7 f4 gxf4 gxf4 exf4 Rxd3+ Ke6 Ra3 Bxc5 Rh1 Ke5 Rxh6 Kxe4 Re6+ Kd5 Re8 a5 Kf3 a4 Kxf4 Kc4 Rxa4+ Kd5 Rd8+ Ke6 Re4+ Kf7 a4 Bb6 Rb8 Ba5 Re5 Bc3 a5 Bd4 a6 Ba7 Ra8 Bb6 a7 Bxa7 Rxa7+ Kg6 Ra6 Kf7 Rxc6 Kg7 Re7+ Kf8 Ra7 Kg8 Rc8#

Game 3

e4 c5 d3 Nc6 Nf3 g6 Be2 Bg7 O-O d6 Be3 Nf6 Nbd2 O-O c3 e5 Rc1 h6 d4 cxd4 cxd4 exd4 Nxd4 Nxd4 Bxd4 Be6 b3 Qa5 Rc2 Rac8 Qc1 Rxc2 Qxc2 Rc8 Qb2 b5 h3 a6 e5 dxe5 Bxe5 Nd7 Bxg7 Qxd2 Qxd2 Kxg7 Bg4 Bxg4 hxg4 Nf6 Qa5 Nxg4 Qxa6 Rc2 Qxb5 Rxa2 Rc1 Nf6 Rc8 Ra1+ Kh2 Ra2 Qb6 Rb2 Qd8

Rxb3 Qh8#

Game 4

g4 d5 Bg2 c6 e3 e5 Ne2 Bd6 h4 h5 g5 Bg4 Bh3 Bxh3 Rxh3 Nd7 d4 e4 c3 Ne7 Nd2 Qc7 c4 dxc4 Nxc4 Bb4+ Kf1 b5 a3 bxc4 axb4 Nd5 Bd2 O-O Rg3 f5 gxf6 Rxf6 Rg5 Raf8 Rxh5 Rf5 Rxf5 Rxf5 f4 exf3 Ng1 f2 Nh3 Rf3 Nxf2 Nxe3+ Bxe3 Rxe3 Qd2 Rd3 Qc2 Rxd4 Rd1 Rxd1+ Qxd1 Qf4 Qxd7 Qc1+ Kg2 Qxb2 Qxa7 c3 Qa8+ Kh7 Qxc6 c2 Nd3 Qd4 Ne1 Qxb4 Qxc2+ g6 h5 Qe4+ Qxe4 Kg7 Qxg6+ Kf8 Qh7 Ke8 Qa7 Kd8 h6 Kc8 h7 Kd8 h8=Q#

Game 5

e4 c5 Ne2 Nc6 d3 g6 g3 Bg7 Bg2 d6 O-O e6 Nd2 Nge7 b3 O-O c4 f5 Rb1 a5 Re1 e5 Bb2 f4 gxf4 exf4 Bxg7 Kxg7 Nxf4 Rxf4 f3 Ne5 Nf1 N7c6 Ng3 Qh4 Rb2 Nd4 Rf1 Bh3 Bxh3 Qxh3 Kh1 Raf8 Rd2 Qh4 Rg1 g5 Qf1 g4 Qd1 Qg5 Rgg2 h5 Rg1 h4 Ne2 Nxe2 Rxe2 h3 Rg3 Rxf3 Re1 Rf2 Re2 Rxe2 Qxe2 Rf3 Rg1 Rxd3 Qe1 Nf3 Qe2 Rd2 Rf1 Rxe2 Rd1 Nd4 Rf1 Nf3 Rd1 Nxh2 Rxd6 Nf3 Rd7+ Kg6 Rxb7 g3 Ra7 Rh2#

Game 6

d4 Nf6 c4 e6 Nc3 Bb4 Bd2 O-O f3 d5 Rc1 b6 cxd5 exd5 a3 Be7 e4 dxe4 fxe4 Bb7 e5 Nd5 Nxd5 Bxd5 Nf3 c5 dxc5 bxc5 Qc2 Nd7 e6 fxe6 Bc4 Nb6 Bd3 c4 Nd4 cxd3 Qxd3 e5 Nf5 e4 Qc3 Bf6 Qc7 Qxc7 Rxc7 Rf7 Rxf7 Kxf7 Rf1 g6 Ng3 Nc4 b3 Nxd2 Kxd2 Rc8 b4 Rc2+ Kxc2 e3 Re1 Bxg2 Rxe3 Bd5 Rd3 Ke6 a4 Be5 b5 h5 a5 h4 Ne2 g5 b6 axb6 axb6 g4 b7 Bxb7 Rb3 Bd5 Rb5 Bf3 Nc1 Bxh2 Rb6+ Kf5 Rb3 g3 Rxf3+ Kg4 Re3 h3 Re1 g2 Re4+ Kg3 Ne2+ Kf2 Kd1 Be5 Rxe5 h2 Rh5 h1=Q+ Rxh1 g1=Q+ Nxg1 Kg3