Frankfurt University of Applied Sciences
Fachbereich 2: Informatik und Ingenieurwissenschaften
Informatik (B.Sc.)

Creating Chess Commentary using Machine Learning

Abstract

This paper deals with the question of how neural networks can be used to create a comprehensive analysis of chess games, which can be used to generate textual, human-understandable, commentary. In particular, we will look at what is needed to represent a chess board that can be used by the neural network to plan and compare moves in order to make an appropriate evaluation of a game of chess. Based on this, we will then explore how the neural network can convert the evaluation into natural language that humans can understand.

Lecturer: Konstantin Ernst
Course: Künstliche Intelligenz und wissenschaftliches Arbeiten
Winter Semester 22/23

Submitted by:
Max Semdner
Matrikelnr.: 1294899
max.semdner@stud.fra-uas.de

Declaration of authorship

I hereby certify that the following project report was written entirely by me and is based on my work unless otherwise indicated. I am aware of the University's regulations regarding plagiarism, including the following actions in the event of a violation. Any form of use of outside work is identified where appropriate and noted in the sources.

Max Semdner

Approved: ____________________

Date: ____________________

Contents

1 Introduction

In the mid-20th century, computer chess experienced its first breakthroughs thanks to the work of scientists like Alan Turing, Claude Shannon and John von Neumann. Alan Turing, the pioneer of artificial intelligence, was convinced that games were an ideal model system for machine learning.[1] This prediction has come true, and machine learning have proven to be an essential part of many chess engines today. In particular, recent projects such as AlphaZero, developed by DeepMind, show how efficient programs which use neural networks are in analyzing board games compared to traditionally used algorithms such as alpha-beta search.[2] Although chess engines have become a powerful tool, they have a lack of transparency regarding the moves they perform. Therefore, professional chess players and commentators are often needed to explain the intention of these moves. This dependence on human chess commentators can be a disadvantage, since moves found by computers can sometimes be misinterpreted or not understood at all. Especially for non-professional chess players these often seem incomprehensible. In what follows, I will explore the question of how this intransparency can be overcome using a neural network to create a virtual chess commentator that uses a built-in chess engine to translate the engine's intentions into a language that humans can understand.

2 Chess Commentator powered by a Neural Network

2.1 General Approach

As described in the next section, each chess engine must meet certain requirements. Based on the requirements, context is generated. Context is a set of properties that apply to the current position on the chess board. These properties are used by the generation models to make comprehensive statements about the moves.

2.2 Chess Engine

Any chess engine must meet a number of requirements in order to function. These requirements include the *representation of the chessboard*, the *prediction of the possible moves* and the *evaluation of the possible moves*. These requirements can be implemented in a number of ways. Certain implementation options have become widely accepted. In the following, one implementation procedure is presented for each requirement. These are state of the art implementations that have been in use for many years or have achieved great success in the recent past.

2.2.1 Board Representation

Since the computer cannot work with a physical chess board and pieces, these must be converted into a form in which the board, pieces, and position can be interpreted by the computer and later used for the input layer of the neural network. The most common form of representation is the data structures bitboards. A bitboard is implemented as an 8×8 array, which is the size of a chessboard. Each array element corresponds to a square on the board. A bitboard is created for each type of piece (pawn, knight, bishop, rook, queen and king) of a given color (black and white). This gives a total number of

[1] Cf. Levy et al. 1982, p. 44-45
[2] See Silver et al. 2018 p. 1

12 bitboards. Finally, the squares on which the figures are placed must be marked on the respective boards. This can be done in binary, where 0 means the square is empty and 1 means the square is not empty.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0
1	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0

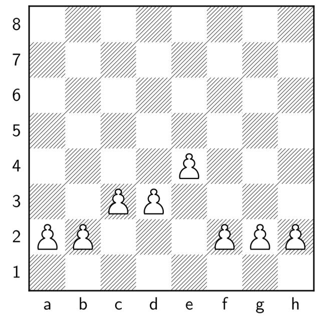

Figure 1: Representation of bitboard with white pawns (left) and the corresponding chessboard (right)

Through the usage of the logical operations, such as AND, OR, NOT the moves can be calculated. An advantage of the logical operations is, that be done quite fast by the processor. Furthermore, with x64 processors the position can be stored in one piece in the memory, as a bit string, since this is exactly 64 bits long due to the number of squares. Another advantage is that due to its simple representation, it can be used as input for the input layer in the neural network without the need to convert it into an understandable format.

2.2.2 Move Prediction and Position Evaluation

A challenge for both humans and computers is to find the best possible move. In fact, chess is considered unsolved, i.e. it is not known if there is an optimal strategy that always leads to victory, for either sides. The objective of a good chess engine is therefore to find the best move based on its computational capabilities. One factor to consider is the depth of analysis. Thus, each move must be considered not only in terms of the current state of the board, but also what effect it will have on subsequent best moves and positions. In order to find the best move, two tasks must be achieved, one is to find the legal, possible moves in the current and the following positions, and the other is the evaluation of these positions.

To implement these two tasks, many systems such as DeepBlue and earlier versions of Stockfish use human-defined scoring functions and game tree search algorithms. The evaluation function receives the board as input and evaluates how good the obtained position is. Game tree search algorithms, such as MinMax, are then used to search for all possible moves (Usually, up to a certain depth, e.g. 15 moves). Each move leads to a new position. Since there is no information about these new positions yet, they are also evaluated by the evaluation function. In the end, the path that has received the highest value from the evaluation function is chosen. Improvements such as alpha-beta pruning remove paths that are known not to yield high values of the evaluation. The problem with this method is that the evaluation functions have to be written by hand with the help of chess experts and constantly refined to achieve an optimal result.

To overcome this problem, and to later get the best results from the engine for the commentator, machine learning is used. Silver et al. (2018) presented an algorithm that solves the two tasks by self-play, using Monte Carlo tree search and a neural network.

Glossary

Stockfish Widespread chess engine. 4

References

Czech, J., P. Korus, and K. Kersting (2021, May). Improving alphazero using monte-carlo graph search. In *Proceedings of the International Conference on Automated Planning and Scheduling, 31*, pp. 103–111. arXiv.

Levy, D. and M. Newborn (1982). *All About Chess and Computers.* Springer Berlin, Heidelberg.

Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Volume 362, pp. 1140–1144.

Zang, H., Z. Yu, and X. Wan (2019, July). Automated chess commentator powered by neural chess engine. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 5952–5961. Association for Computational Linguistics.