Jakeb Milburn

CISC 484

8/11/23

Predicting Chess Win-Rate based on Sequence of Moves using RNN

Introduction

This research delves into predicting chess match outcomes through analyzing move sequences with Recurrent Neural Networks (RNNs). By tapping into RNNs' sequential processing abilities, we aim to uncover hidden correlations, patterns, and strategies woven within chess games.

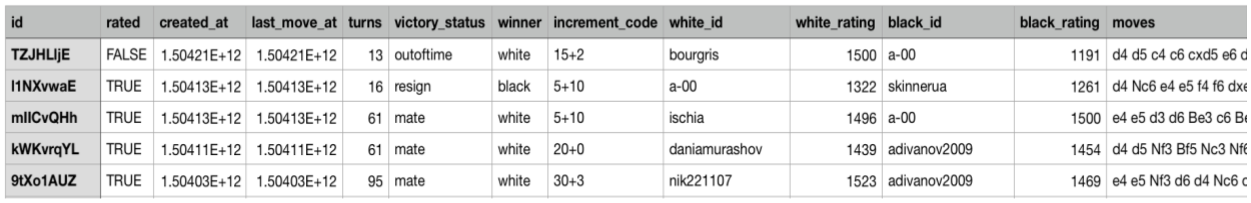
The fusion of chess and RNN technology aims to predict match outcomes based solely on players' move sequences. Yet, our study's implications aren't confined to theory. It has the potential to reshape chess openings and mid-game strategies in the real world.

This paper methodically details move preprocessing and RNN model tuning. It delves into tokenization to better format data and methods to increase model accuracy on such a dataset. By extracting insights from one of the most popular online chess platforms, we quantify the predictive capabilities inherent in chess move sequences, offering a structured framework to comprehend the game's dynamics.

Our study offers practical applications. It could redefine chess openings, aiding players in crafting strategies tailored to potential outcomes. Additionally, predicting win-rates mid-game presents a way to give viewers of high-level games an idea of who is winning. Marrying traditional chess strategies with modern RNN capabilities, we could unveil insights within move sequences, enriching both chess theory and real-time gameplay analysis.

The Dataset

The dataset, sourced from the popular online chess platform Lichess.com found on Kaggle, holds a comprehensive collection of over 20,000 chess games, structured in a tabular format, with each row representing a distinct game and capturing datapoints of chess dynamics including attributes such as "winner," "move sequence," "time increment," and more. In this dataset, our focus for the purpose of this experiment is: the categorical "winner" classification and the "move sequence" attribute, set up as a list of moves in the format “e4 e5 kf6” for example. Serving as the foundation for our research, this dataset forms the groundwork for training and evaluating our predictive model driven by RNNs where we aim to discern the correlation between "move sequence" and "winner”, ultimately contributing to both the predictive domain of chess and the broader discourse on the application of deep learning in complex strategic contexts.



id	rated	created_at	last_move_at	turns	victory_status	winner	increment_code	white_id	white_rating	black_id	black_rating	moves
TZJHLljE	FALSE	1.50421E+12	1.50421E+12	13	outoftime	white	15+2	bourgris	1500	a-00	1191	d4 d5 c4 c6 cxd5 e6 d
l1NXvwaE	TRUE	1.50413E+12	1.50413E+12	16	resign	black	5+10	a-00	1322	skinnerua	1261	d4 Nc6 e4 e5 f4 f6 dxe
mIlCvQHh	TRUE	1.50413E+12	1.50413E+12	61	mate	white	5+10	ischia	1496	a-00	1500	e4 e5 d3 d6 Be3 c6 Be
kWKvrqYL	TRUE	1.50411E+12	1.50411E+12	61	mate	white	20+0	daniamurashov	1439	adivanov2009	1454	d4 d5 Nf3 Bf5 Nc3 Nf6
9tXo1AUZ	TRUE	1.50403E+12	1.50403E+12	95	mate	white	30+3	nik221107	1523	adivanov2009	1469	e4 e5 Nf3 d6 d4 Nc6 c

Representation of the dataset

Challenges

Predicting chess win-rates using RNNs is a formidable task, largely owing to the inherent intricacies of the game. Chess's strategic landscape, characterized by limitless potential moves and intricate decision-making, complicates the extraction of predictable patterns solely from move sequences. The way that one move can change an entire game further amplify this complexity, with single moves ending in consequences that are challenging to model accurately within an RNN framework.

However, the complexity is not confined solely to the nature of the game. The data itself, drawn from an online platform where players can prematurely resign or abandon matches time, introduces a layer of imperfection. This variability adds noise to the dataset, as even well-played games might be ended unexpectedly, impacting the accuracy of predictive modeling. Furthermore, this data originates from diverse players with varying skill levels, strategies, and intentions, increasing the challenge of isolating meaningful patterns that lead to reliable win-rate predictions.

Furthermore, the data processing phase emerges as a hurdle. The way that the move sequences are listed are challenging to feed into a model, making pre-processing a crucial step. This phase assumes a pivotal role as it shapes the quality of the input fed into the RNN, directly influencing the model's capacity to extract meaningful insights.

Related Work

Various machine learning solutions have been explored to enhance chess engines. Convolutional Neural Networks have been used to predict move quality and guide search strategies. Reinforcement Learning, seen in projects like AlphaZero, lets engines learn from self-play to improve move selection. Additionally, Monte Carlo Tree Search (MCTS) has been combined with machine learning techniques to optimize move selection and expand search trees efficiently.

Our decision to employ RNNs for predicting chess win-rates is influenced by prior research, specifically a study that investigated the chess knowledge development within the MuZero neural network, a powerful chess engine that is much better at chess than any human.

In their research, MuZero consists of two main components: a recurrent neural network that calculates the probability distribution for the next move and the anticipated outcome of the game based on a given state (current board position, move history, and additional game-specific information), and a Monte Carlo tree search (MCTS) module that repeatedly evaluates states and guides decision-making.

We chose to only utilize the RNN for the purpose of calculating the outcome of the game from the move sequence, as other methods would have been overkill since we are not trying to calculate the next move.

Approach

Predicting win rates from a sequence of chess moves using RNNs is essentially a classification problem. In this context, the objective is to categorize the final outcome of a chess game based on the sequence of moves played by both players, so the answer is either white or black.

We used a TensorFlow Keras RNN with Gated Recurrent Unit (GRU) architecture, utilizing an Adam optimizer and Cross Entropy loss function for this experiment

GRU is well-suited for sequential data like chess moves. It can effectively capture long-range dependencies in move sequences. The gating mechanisms in the GRU enable it to mitigate the vanishing gradient problem, ensuring gradients flow effectively during training.

The Adam optimizer’s adaptive properties allow the model to adjust learning rates for each parameter, improving convergence speed and avoiding potential pitfalls of fixed learning rates.

The choice of Cross Entropy quantifies the difference between predicted and actual class probabilities, guiding the model to minimize misclassifications. Given the nature of the task the Cross Entropy loss effectively drives the RNN to learn strategic patterns that lead to different game outcomes.

However, modifications were needed for the data first in order to feed it into the model. We used a tokenizer to process the data before it was fed into the RNN. The tokenizer acts as an intermediary step that converts the textual move sequences into a numerical representation that the RNN can process effectively. Initially, the raw string of chess moves is parsed into individual moves. Each move is then encoded into a unique token. The tokenizer maintains a vocabulary that maps each unique move to a corresponding numerical token. During the encoding process, the tokenizer replaces each textual move with its corresponding token. The data was then ready to be accessed by the RNN.

Training and Testing

For this project, the dataset was divided into training and testing subsets using a 70/30 train-test split. Furthermore, to monitor model performance and prevent overfitting, a validation split of 20% was set aside from the training data.

The model was trained for a total of 3 epochs, with each epoch encompassing a complete iteration through the training data. A batch size of 32 was employed, allowing the model to process 32 sequences of moves in each training step. The Adam optimizer was utilized to optimize the model's parameters during training, as previously mentioned.

As previously mentioned, the data processing phase played a crucial role in this project by converting raw chess move sequences into a numerical format suitable for the RNN. We further processed the data by eliminating all instances of draws, removing unnecessary games that did not lead either side to a win. We did not process the data past this point because, while there are certainly games in the dataset that are outliers that may have affected the model’s performance, it is likely that some of these outlier games contained sequences that statistically lead to a win, and it is impossible to fully prune out every game that decreases the model’s performance.

The primary evaluation metric used to assess the performance of the model in predicting win rates from chess move sequences was accuracy. This metric provides a clear indication of how well the model's predictions align with the actual outcomes of the games. A high accuracy score signifies that the model is effectively capturing and learning from the patterns within the move sequences. By quantifying the model's correctness in predicting win rates, accuracy serves as a reliable measure of the model's overall performance in this classification task.

Results

The results of this project indicate the performance of model in predicting win rates from chess move sequences across three epochs of training was successful. In the initial epoch, the model achieved an accuracy of 73%, which suggests a moderate ability to predict game outcomes accurately. However, the corresponding loss value of 0.54 indicates room for improvement in minimizing the discrepancy between predicted and actual outcomes.

As the training progressed to the second epoch, a notable improvement in accuracy was observed, reaching 85%. This increase suggests that the model is effectively learning and capturing more intricate patterns within the move sequences, leading to more accurate predictions. Correspondingly, the loss value decreased to 0.35, indicating that the model's predictions are aligning more closely with the actual outcomes.

In the final epoch, the model's accuracy slightly decreased to 80%, yet it still demonstrated a commendable ability to predict game results. The loss value also increased slightly to 0.44, but it remained at a level indicating effective learning and convergence.

The observed decrease in accuracy during the final epoch of training could be attributed to several factors. One likely factor is overfitting, the model may have become too specified on the limited training data and lost its ability to generalize to new, unseen data.

Another possible reason is the complexity of the dataset. Chess move sequences are inherently intricate, containing many strategic possibilities. As the model learns from these sequences over multiple epochs, it might encounter complexities that are challenging to capture, especially with a limited number of training iterations.

The final reason could be the noise of the dataset. This noise arises from various sources, including player behavior, varying skill levels, strategic choices, and unpredictable outcomes. Players might make suboptimal moves or deviate from conventional strategies, introducing irregularities that the model needs to account for. Additionally, differing skill levels among players contribute to the complexity of the dataset, as moves that might be perceived as noise by the model could be intentional strategic decisions by players of varying expertise.

An ablation study was not conducted in this project primarily due to the nature of the dataset and the simplicity of the features. The only feature utilized in the analysis was the sequence of chess moves itself. Given that our features consisted solely of move sequences, conducting an ablation study would not have provided meaningful insights.

In conclusion, the model showcased its ability to capture nuanced patterns within the sequences, enabling accurate predictions of game outcomes. While fluctuations in accuracy across epochs were observed, the overall trend reflected the model's progressive understanding of strategic dynamics. As the project demonstrates, the integration of machine learning techniques with chess analytics offers a promising avenue for understanding the correlation between move sequences and game results, contributing to the broader landscape of strategic decision-making and predictive modeling.

References

Omid, D. (n.d.). DeepChess: End-to-end deep neural network for automatic learning in chess. https://www.cs.tau.ac.il/~wolf/papers/deepchess.pdf

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., & Silver, D. (2020, February 21). *Mastering Atari, go, chess and shogi by planning with a learned model*. arXiv.org. https://arxiv.org/abs/1911.08265v2

MacGrath, T. (n.d.). Acquisition of chess knowledge in alphazero | PNAS. https://www.pnas.org/doi/10.1073/pnas.2206625119

J, M. (2017, September 4). *Chess game dataset (lichess)*. Kaggle. https://www.kaggle.com/datasets/datasnaek/chess

Pai, A. (2023, July 24). *What is tokenization in NLP? here’s All you need to know*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/