Classification of PGN

Logan Miller
Departments of Math and Computer Science
Lawrence Technological University
Southfield, MI, USA
lmiller2@ltu.edu

Rory Wilson
Department of Math and Computer Science
Lawrence Technological University
Southfield, MI, USA
rwilson2@ltu.edu

***Abstract*—Make this a summarization of your paper.Make this a summarization of your paper.Make this a summarization of your paper.Make this a summarization of your paper.Make this a summarization of your paper.Make this a summarization of your**

***Index Terms*—PGN, Clustering, Chess.**

I. Introduction

Chess is one of the oldest and most universally played strategy games, and with the advent of online platforms like chess.com, millions of games are now available for analysis. These platforms generate a wealth of data, which presents an opportunity to gain insights into player behavior, skill levels, and strategies. However, analyzing such large datasets to extract meaningful patterns remains a challenge. While traditional methods often focus on broad metrics like Elo ratings, they fail to account for the nuanced differences in play styles and strategies that define various skill levels. This paper aims to address this gap by employing machine learning techniques, specifically clustering, to categorize chess games based on player skill levels and game play characteristics, offering a more granular understanding of player performance.

The importance of this problem lies in its potential to enhance player development, improve training tools, and refine chess engines. Automatically classifying games by skill level can provide valuable feedback to players, enabling personalized improvement suggestions. Additionally, it offers coaches and analysts a more detailed view of game play, which can inform training strategies. The key challenges in this task include handling the complex, high-dimensional nature of chess data and identifying relevant features that effectively differentiate skill levels. The main contribution of this work is a novel clustering approach that groups games based on Elo ratings, move length, and game result, providing deeper insights into the relationship between game play and player expertise.

II. Related Work

- Previous approaches to the problem
- Relevant methods and techniques
- Limitations of existing work
- How your work differs/improves upon prior work

III. Methodology

A. Data Collection and Preprocessing

To gather the data about the different chess games there is a website https://database.lichess.org/ that is filled with different games with different levels of players. the data includes the move sequences, player ratings, game outcomes, and other metadata, which provides the foundation for clustering analysis. The data is cleaned by removing games with non-standard variants, incomplete results, or missing Elo ratings.

The preprocessing steps include splitting each game's metadata and move sequences, then filtering out games that do not meet the following criteria:

- The game must be a standard chess game (no variants).
- The game must have a normal termination and result.
- The game must have valid Elo ratings for both players.

The cleaned data consist of three key features for each game: the White player's Elo rating, the number of moves made during the game, and the game results (encoded as 0, 0.5, or 1 for black win, draw, or white win, respectively)

Portable Game Notation (PGN) text comes in the form of several tags, each a key-value pair enclosed within square brackets, and a list of moves with a counter of how many full turns have passed. Since the text is already mainly of the form of key-value pairs, we read each one into a dictionary, with the moves coming after being added with the key "Moves."

The second step is to isolate the data we need from the rest: The Elo ratings of the players, the result of the match-up, and the list of moves. While the Elo ratings and the result are their own tags and can be easily isolated, the moves require a bit of cleaning to be usable: 1. We remove all the turn counters so that only the move text is left. 2. Comments, denoted by curly braces, are for human eyes only, and are also removed. 3. Any other text which is not part of a move is removed with a regular expression substitution. This removes additional information like the exclamation points on brilliant moves and the question marks on blunders, to maintain the purity of the text. 4. Any extended lengths of whitespace left behind are condensed to one.

As an additional angle to attempt, we use the chess-python library to also convert these PGN moves to UCI moves, which are only denoted by the square moved from concatenated with the square moved to.

Every game is also condensed into a simple tuple of data to run additional measurements upon, consisting of just the White player's Elo rating, the length of the game in full moves, and the result of the game.

B. Feature Extraction

The feature extraction process focuses on simplifying and organizing the data for use in clustering algorithms. The extracted features include:

- Player Elo ratings: The rating of the white and black players
- Move length: The total number of moves made during the game.
- Game result: The outcome of the game, is represented numerically for clustering purposes.

The moves are split into individual actions, and each game's result is encoded to represent the player's performance, either as a win, loss, or draw.

For the purposes of measurement, Elo ratings are grouped into bins of width 100, with e.g. the 700s bin holding all games where the White player had an Elo rating between 700 and 799, inclusive. We record the number of games in each bin, as well as how many were wins, draws, or losses.

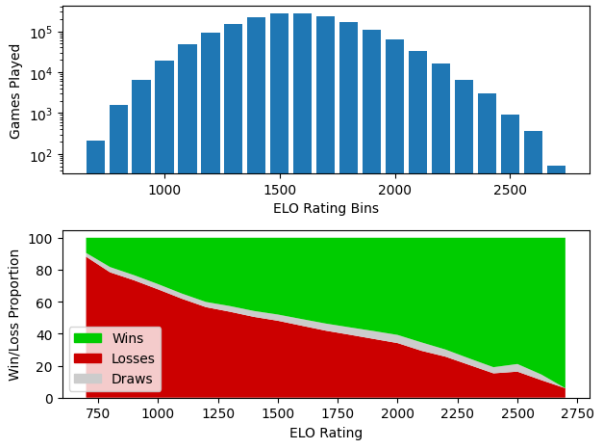

Fig. 1. Games Played vs. ELO Rating Bins (top): A bar chart showing the number of games played in each ELO rating range. Win/Loss/Draw Proportions vs. ELO Rating (bottom): A stacked area chart visualizing the proportions of wins (green), losses (red), and draws (gray) for each ELO rating range.

In terms of the distribution of games, it appears to closely follow a normal distribution (or a unimodal distribution of slight positive skew). Games in the range of 1400 to 1799 make up approximately 58% of the dataset. There is a very clear positive correlation between Elo rating and win percentage, indicating that players with high Elo ratings win more of their games.

C. Clustering Implementation

The cleaned dataset is used to group games based on player performance. The program applies the k-means clustering algorithm, leveraging Elo ratings as the primary axis for separation. Key steps include:

- Determining the number of clusters(k): Using silhouette scores, the optimal value for k is selected, ensuring well-defined groupings.
- Running k-means: the algorithm partitions the data into clusters, where each cluster represents a subset of games with similar Elo ranges and gameplay characteristics.

D. Statistical Analysis and Visualization

After clustering the dataset, statistical analyses and visualization are performed to interpret the relationship between Elo ratings, game results, and gameplay patterns. these analyses highlight trends across different skill levels and provide actionable insight into the dataset.

IV. Experimental Setup

A. Dataset

Original data can be accessed under the open database at Lichess.org. This is a very good source of raw record data for chess games, such as several billion games within the Lichess platform. It maintains metadata about the players' ratings, time controls, and outcomes, along with full move sequences in PGN format. The database is a treasure for research due to its size, diversity, and good accessibility; material on most types of games and all skill levels from beginners through grandmasters is covered.

In this project, only a subset of the data will be selected to demonstrate how some standard features like Rating, Total Move Count, and Outcome enable some effective clustering analysis. The data is already in structured PGN format; therefore, parsing and feature extraction will be easier. It will be ideal for exploring patterns and trends occurring in the play of chess.

This is documented in greater detail below, especially in Table 1, which outlines the structure and content of the files that make up the dataset.

TABLE I

lichess.org dataset				
White name	Black name	Result	White Elo	Black Elo
paul2chess3	Andique	1-0	1509	1623
Shenhewho	Gentux	1-0	1857	1963

The table above shows 2 games from the dataset showing the payer names, the result of the game, and each player elo. there is other data included within the file but most of it is nonsense.

B. Evaluation Metrics

Quality clustering for the PGN project was made using some different metrics; among them, the most important was the silhouette score since this score gives a measure of how well the clusters are kept apart, and how well the game fits within its assigned cluster. In that way, clusters reflect real differences in either the level of players or their style of play. Additionally, inertia (within-cluster sum of squares)

was used to evaluate the compactness of data points within clusters. Cluster size distribution was analyzed to identify any imbalances or outliers affecting the clustering process. To ensure the clusters' interpretability, we compared results against logical groupings, such as player rating ranges or typical game lengths, confirming that the clusters provided insight into game dynamics. These also allow us to assess the performance of our approach against overall simpler baselines such as rating-only clustering.

C. Baseline

The baselines were necessary for giving a comparison to the effectiveness of the clustering methodology on PGN data. A simple baseline entails the grouping of chess games regarding players' ratings into pre-defined ranges of cut-off, for instance, beginning, intermediate, and advanced players. This was a simplified comparison that didn't give a deep scope on how other features such as game length and its outcome interact. Other baselines were k-means clustering on fewer features for further benchmark performance checks. Comparing those with our multi-feature clustering methodology, we could then illustrate the advantages of integrating diverse features for nuanced gameplay. Since we were on the free version of Google Colab, runtime and memory needed to be optimized for baseline testing to do efficient experiments within resource bounds.

D. Implementation Details

The PGN clustering project has been developed in Python using the libraries: numpy for numerical operations, matplotlib for visualization, and sklearn to apply k-means clustering. For PGN elaboration, a personal parser has been used to extract relevant features from each match: players' ratings, total moves, and game outcomes. Then, data normalization was performed for correct clustering, while missing values were treated by deleting incomplete games. Therefore, k-means implementation was employed for the structured dataset, while the number of clusters was determined by the elbow method and silhouette score for the best performance. All the computations have been done on the free version of Google Colab, where computational resources were good enough to run this clustering analysis, though sometimes very limiting regarding duration and hardware.

E. Hyperparameters

The hyperparameters in this program are tuned carefully to optimize the clustering and feature extraction performance of the chess games. For the k-means clustering algorithm, the most important hyperparameter was the number of clusters derived from the unique groups of ELO rating determined by the program. This made sure that the grouping of the skill levels of players was logical. We varied the RESOLUTION parameter from this model to 500, which grouped the ELO ratings into bins of 500, thus affecting clustering granularity. The preprocessing feature extraction also went through a TF-IDF vectorizer where, among its parameters, tokenization and stopword filtering were developed focusing on dealing with the densest pieces of text data. These hyperparameters control parsing data, such as GAME LIMIT, but also optional flags like DOUBLE GAMES. These balance the volume and variety of preprocessed data for scalability but still meaningful results. These have been iteratively tuned through testing to balance performance with computational efficiency.

F. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 1", even at the beginning of a sentence.

Before attempting any sort of classification, we first remove the top and bottom 1% of data for the purpose of removing any outliers. Games are then grouped by Elo rating into groups of 500, normalized so that the group containing the least experienced players is group 0.

We use a combination of a variety of classification techniques on both the PGN moves and the UCI moves to see which are best suited for the task. Each combination will be represented with a set of 3 symbols, to be appended to the document set used. - We compare and contrast two kinds of vectorizers. The first is a TF-IDF vectorization, with the idea being that players of a certain skill level will tend towards certain moves which can be used to distinguish them. The second is a vectorization to word counts as a baseline. It's important to make sure that the vectorizer does not force the text to lowercase; It doesn't matter with UCI, but PGN text has integral capitalization which needs to be preserved. This toggle is the first symbol of the set; TF-IDF vectorization will use T as its symbol, and word count vectorization will use C. - The data then needs to be split into a training group and a test group. 75% of the data will be used to train the classifiers, and the remaining 25% will be used to test their accuracy. The imbalance of the initial dataset is exaggerated with the increase in group size, so we additionally see how stratifying the data so that each class is equally represented affects the accuracy. If the data is stratified, the second symbol will be 1, and if not, it will be 0. - Two kinds of Naive-Bayes classifiers are used to attempt to model and predict the data. First is a Multinomial Naive-Bayes (MNB) classification, and the second is a Complement Naive-Bayes (CNB) classification. CNB is designed to take the complement of each class for its weights, and is suited to imbalanced datasets. Theoretically, CMB will perform better than MNB due to the imbalance of the dataset it will be given. MNB will be M, CMB will be C.

As an example, UCI text which is treated to a TF-IDF vectorization, stratified data, and Complement Naive-Bayes classification will be represented as "UCI-T1C."

V. Results and Discussion

PGN-T1M: 62.232389% PGN-T1C: 55.505847% PGN-T0M: 62.264947% PGN-T0C: 55.520559% PGN-C1M: 57.755987% PGN-C1C: 55.037731% PGN-C0M: 57.753334% PGN-C0C: 55.062090%

UCI-T1M: 62.297264% UCI-T1C: 55.598699% UCI-T0M: 62.330305% UCI-T0C: 55.622575% UCI-C1M: 56.749333% UCI-C1C: 54.837317% UCI-C0M: 56.740892% UCI-C0C: 54.786912%

A. Overall

On PGN text, the highest accuracy is 62.26%, with the Multinomial Naive-Bayes classifier, unstratified data splits, and the TF-IDF vectorizer (PGN-T0M). These same settings produce the highest accuracy for UCI text as well, at 62.33% (UCI-T0M).

B. Trends

Some trends appear in regards to how specific choices affect the average accuracy of the predictions. - Generally, classification is improved by using TF-IDF vectorization over word count vectorization, with MNB losing approximately 5-6% accuracy when switching between the two and CMB losing 0.5-0.9%. - Strafitication affected accuracy very little, with the random sampling picking approximately the same frequency of classes as the guaranteed frequencies enforced by stratification. In most cases, not using stratification increased overall accuracy by a few hundredths of a percentage point, except when using UCI text with word count vectorization, in which case accuracy decreases approxumately 0.01% with MNB and 0.05% with CMB. - Against expectations, Multinomial Naive-Bayes outperforms Complement Naive-Bayes in all cases, being approximately 7% more accurate with TD-IDF vectorization and 2% more accurate with word count vectorization.

VI. Conclusion

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

References

[1] F. Wijayanto, “Clustering Analysis of Chess Portable Game Notation Text,” Jurnal Sains, Nalar, Dan Aplikasi Teknologi Informasi, vol. 3, no. 3, pp. 137–142, 2024.

[2] K. Raghav and L. Ahuja, “Chess Opening Analysis Using DBSCAN Clustering and Predictive Modeling,” in 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, Mar. 14–15, 2024, pp. 1–5.

[3] C. Bauckhage, A. Drachen, and R. Sifa, “Clustering Game Behavior Data,” IEEE Transactions on Computational Intelligence and AI in Games, vol. 7, no. 3, 2015.

[4] S. Shukla and S. Naganna, “A review on K-means data clustering approach,” International Journal of Information & Computation Technology, vol. 4, no. 17, pp. 1847–1860, 2014.

[5] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, “An efficient K-means clustering algorithm: Analysis and implementation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp., 2002.

[6] B. S. Kong, I. Hipiny, and H. Ujir, “Classification of digital chess pieces and board position using SIFT,” in 2021 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala Terengganu, Malaysia, Sept. 13–15, 2021, pp. 66–71.

[7] D. Shamsudin, L. M. Chew, and O. L. Yeng, “Clustering algorithms analysis based on arcade game player behavior,” in 2022 IEEE Fifth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), Laguna Hills, CA, USA, Sept. 19–21, 2022, pp. 122–125.

[8] H. Kwon, W. Jeong, D.-W. Kim, and S.-I. Yang, “Clustering player behavioral data and improving performance of churn prediction from mobile game,” in 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea (South), Oct. 17–19, 2018, pp. 1252–1254.

[9] H. Wang, X. Liu, and M. Shi, “Study on the chessboard recognition and positioning method of Chess Game System,” Advances in Intelligent Systems Research, 2015.

[10] X. Liu, J. Jin, Q. Zhong, Q. Lei, Z. Wu, and Z. Fang, “Vision-based chess detection for a robotic companion,” in 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Chengdu, China, Apr. 23–26, 2021, pp. 248–253.