Revealing Game Dynamics via Word Embeddings of Gameplay Data

Younès Rabii, Michael Cook
Queen Mary University of London
{younes, mike}@knivesandpaintbrushes.org

Abstract

In this paper we show how word embeddings, a technique used most commonly for natural language processing, can be repurposed to analyse gameplay data. Using a large study of chess games and applying the popular Word2Vec algorithm, we show that the resulting vector representation can reveal both common knowledge and subtle details about the game, including relative piece values and the natural spatial flow of chess play. Our results suggest that word embeddings are a cheap and simple technique that can provide a broad overview of a game's dynamics, helping designers and critics form new hypotheses about a game's design, structure and flow.

Introduction

Analysing a game's design is a vital part of developing, improving and critiquing games. As a result, there are many different approaches to game analysis, originating in different disciplines, and focusing on analysing different aspects of a game. In order to analyse a game's feel or aesthetics, for example, we might perform an in-person playtest and survey players (Gow et al. 2012). If we want to analyse game balance, we could log data from testers and apply statistical methods to analyse patterns (Giovannetti 2019).

Most analytical techniques are designed to answer very specific questions, and also require us to have already formed hypotheses that we wish to test. For example, playtesting uses carefully-designed questionnaires to answer particular questions about the play experience. Game balance analysis records and analyses specific data to see if they match a specific definition of game balance the designer is interested in. However, it is not always possible to form hypotheses in advance. If we are early in a game's development, joining a project we are unfamiliar with, or are not an experienced designer, we may struggle to know what questions to ask, or where to go looking for them.

These techniques can also be very narrowly focused on specific questions. This is what makes them so powerful when used correctly, but it can also make them limiting. As Giovanetti puts it in (Giovannetti 2019), 'sometimes other factors can cause the data to lie to you'. In developing the card game *Slay The Spire* his data-driven analysis suggested a particular card was too powerful as its winrate was very high. However, upon closer examination it was only offered near the end of the game, and so was mostly encountered by good players who were already close to winning.

In this paper we present a new way of analysing games, by repurposing a natural language processing technique called *word embeddings* to analyse gameplay data. This technique is easily applied to data with little preparation, and can be visualised in easy-to-read two-dimensional matrices that swiftly reveal complex and nuanced relationships between game concepts. The resulting analysis provides a big-picture overview of many different parts of a game's design without a narrow focus on one area, and without any detailed hypotheses about the design itself. As a result, it not only provides useful knowledge and insight, but it also helps a user formulate hypotheses and identify areas for future analysis.

We demonstrate our approach in this paper by applying it to chess, and show that it is able to rediscover complex game knowledge such as the accepted relative value of different pieces or the relationship between special game actions such as castling, simply by parsing and analysing game events as natural language. In many cases the analysis revealed dynamics that were novel to the authors as beginner chess players, which led us to our own follow-up investigations to confirm them. We believe this shows how the technique can support a high-level overview of a game's dynamics, as well as supporting the formation of new hypotheses and further analysis. Furthermore, we report preliminary results that suggest that this technique can be used on AI play logs as well as human players, and is sensitive to player skill.

The remainder of this paper is organised as follows: in *Background* we introduce word embeddings and some aspects of chess and survey related work; in *Creating the Model* we describe how we apply word embeddings to gameplay logs, including data preprocessing; in *Using the Model* we report on applying our method to 250,000 games of chess, showing how it can be interpreted and queried; in *Future Works* we discuss further lines of inquiry comparing average human player data to AI agents and expert players; finally, in *Conclusions* we summarise our contributions.

Background and Related Work

Word Embeddings and Word2Vec

A word embedding is a way of representing a corpus of words in a high-dimensional vector space, such that words which appear in similar contexts in the corpus are closer together in the vector space. As a result, distance in the vector space becomes a proxy for semantic similarity, allowing us to measure how similar two words are by how far apart they are in a word embedding. Because words are represented as vectors, this allows us to perform other operations on them, such as adding or subtracting vectors from one another. For example, a word embedding constructed on a corpus of text might allow us to calculate $\mathrm{Paris}-\mathrm{France}+\mathrm{Morocco}$, yielding Rabat, the capital of Morocco. The subtraction extracts the relationship 'capital of', and the addition applies that relationship to Morocco, giving us our result.

Word2Vec is a popular algorithm for creating word embeddings, and has been widely studied in the NLP community and beyond (Mikolov et al. 2013). Starting from a textual corpus composed of words, Word2Vec creates a mapping from each token in the vocabulary of the corpus to a scalar vector (Řehůřek and Sojka 2010). Word2Vec's power and simplicity has led it to have impact outside of academic research, used by artists, developers and the general public (Parrish 2018).

Chess

Chess is a popular board game played in its earliest forms since the 6th century (Murray 1986). Its popularity can be partly explained by how easy it is to understand its rules compared to how complex it is to master its play. In the last ten centuries, chess has been studied by various scholars (Al-Suli 941; Murray 1986), generating a lot of expert knowledge around it – most of which, cannot trivially be derived from simply reading the rules.

Related Work

In (Zhong, Nakashima, and Akiyama 2019) the authors use Word2Vec to analyse different teams of robots entering the RoboCup robot soccer tournament. Additional logging is added to the system to record the actions robots take, such as passing the ball, and team performances are then analysed in order to try and measure the similarities between different team playstyles. Word2Vec is used in this work primarily as a means to relate one dataset to another, whereas we primarily use Word2Vec as a tool to analyse a single dataset. However, we will also compare between datasets later in the paper as a way to illustrate points about our technique.

A wide variety of work in automated game design uses metrics to analyse games as they are being designed and tested. In (Browne and Maire 2010) the authors use a set of over 50 hand-designed metrics to evaluate different qualities of a game, for example, including decisiveness and drama. In all of these cases, the metrics represent prior knowledge from the system's creators about the desired results of the system, and focuses primarily on measuring single-dimensional properties of the game such as winrate. Our analysis does not require the user to have any prior assumptions or beliefs about their design, and rather than focusing on a specific feature it provides a broader overview of the game as a whole. We believe these two approaches would work very well in tandem with one another.

There is also evidence that some tools designed for analysing game systems can become hard for users to work with if they require specific analytical metrics to be defined in advance. In (Cook et al. 2019) the authors describe a tool for analysing spaces of generative content. Although the tool is designed for exploratory use by non-experts, it requires some metrics to be defined before use, which many users struggled to do. This shows the usefulness of more general exploratory techniques like our approach, which can help analyse game systems in order to help the formation of more specific hypotheses and metrics for future analysis.

Creating the Model

In this section we describe the process of preparing a dataset, applying Word2Vec to create a model, and subsequently analysing the resulting model for insights into the game.

Data Gathering

First, we must collect data to analyse, and ensure it contains enough information to provide Word2Vec with meaningful relationships to model. Our worked example uses chess game logs, because the game's popularity means game logs are readily available in large quantities, and the game is well-studied enough that we can support our analysis with research carried out by others. We gathered a dataset of 250,000 chess games from the Lichess Open Database[1], which stores games played on the popular online chess service Lichess. Lichess has hosted over 2 billion games of chess since 2010. We sampled our dataset of 250,000 games from 2013, covering all skill levels, to simulate a broad sample of playstyles from a snapshot in time.

Game logs recorded by Lichess are written in the *Portable Game Notation* (PGN) format (Steven J. Edwards 1994). This notation format is a kind of shorthand for recording chess games in a compact fashion. However, this compact nature also means a lot of information is omitted, and relies instead on minimally describing the changes caused by each player action. For example, the PGN notation for a specific chess move often omits which piece is moving unless there is an ambiguity, and never mentions which pieces are captured when a capture is made. This information can be deduced by re-enacting the game while reading the log, but is omitted in its textual representation.

Our approach uses Word2Vec to extract contextual relationships between data, as if the game logs were a language of their own. Thus, omitting information such as which piece is being captured is like removing the subject from a sentence – a human might be able to intuit it from context, but the sentence loses a lot of information. After some initial experimentation with the PGN format, which yielded poor results, we decided to preprocess the data by converting it into a more verbose language, which adds in some of this

[1] https://database.lichess.org/

`Turn Black Rook C4 R7 C4 R0 Capture Knight`	*Black moved a Rook from (4,7) to (4,0), capturing a Knight.*
`Turn Black Pawn C5 R1 C5 R0 Promote Queen Check`	*Black moved a Pawn from (5,1) to (5,0), promoting it to a Queen. The game is in check.*
`Turn White King C4 R7 C2 R7 Castling Checkmate WhiteWin`	*White moved its King from (4,7) to (2,7), with the single-use castling move. The game is in checkmate. White wins.*

Table 1: Excerpts of chess games in our description language (left) and their natural language equivalent (right)

implicit data, so that each line of the game log contains a complete description of the action taking place in the game.

Data Preparation

The custom description language we designed allows us to describe each game with a series of tokens, where each token is mapped to a concept in chess. Each game log begins with the `Start` token and ends with the `End` token. For each turn of the game, we add a new line of tokens with the syntax:

`Turn [Player] [Piece] [Start] [Dest] [Actions] [State]`

Where `[Player]` is the player moving this turn; `[Piece]` is the chess piece moved; `[Start]` and `[Dest]` are two sets of coordinates indicating where the piece was before and after its move; `[Actions]` indicate which additional actions were done in the turn (e.g.: capturing a piece); and `[State]` indicate which state the game is in after the move (e.g.: Checkmate). A sample of logs produced by this conversion process can be found in Table 1, along with their natural language equivalent. A comprehensive list of all tokens we defined for chess can be found in Table 2.

Configuring and Applying Word2Vec

For the models described in this paper, we used the standard open-source Gensim implementation of Word2Vec (Řehůřek and Sojka 2010). We used the default settings for Word2Vec with two exceptions: the size of *context window* and the number of *dimensions* in the model, which we changed to reflect the difference between the default application domain (English text) and our application to chess.

The size of the context window was increased from the default of 5 words to 20. A small context window is well-suited to the English language, where the vocabulary size is extremely large (in the order of 10^6 tokens for a large dataset) and a model can easily learn context from a handful of adjacent tokens (Mikolov et al. 2013). In our dataset, with just 36 tokens in the vocabulary, the model benefits from more context to understand the relationship between tokens.

The number of dimensions in the resulting model was reduced from 100 to 5. Each token in the resulting model is represented by a vector of a size equal to the number of dimensions, therefore fewer dimensions means a token is represented by a smaller vector. With only 36 tokens in our vocabulary, too many dimensions would allow Word2Vec to overfit and produce a model with no meaningful relationships or information in it. For English language datasets with very large vocabularies, 100 dimensions is still a challenge for the algorithm. We chose just 5 dimensions for our model, with the reasoning being that this is approximately $log_2(36)$, and thus the number of bits required to represent a vocabulary of 36 values. Preliminary experimentation with this was positive enough for us to continue.

Token	Corresponding Chess Concept
Temporal Concepts	
`Start`	Beginning of the game
`End`	End of the game
`Turn`	Beginning of a turn
Spatial Concepts	
$\texttt{R}_i$	Row number i, with $i \in [\|0; 7\|]$
$\texttt{C}_i$	Column number i, with $i \in [\|0; 7\|]$
Piece Concepts	
`Pawn`	The "*Pawn*" chess piece
`Rook`	The "*Rook*" chess piece
`Knight`	The "*Knight*" chess piece
`Bishop`	The "*Bishop*" chess piece
`Queen`	The "*Queen*" chess piece
`King`	The "*King*" chess piece
Action Concepts	
`Capture` $[P]$	The capture of piece $[P]$
`Promote` $[P]$	The promotion of a pawn to piece $[P]$
`Castling`	The single-use "Castling" move
Game State Concept	
`Check`	The game is in "Check"
`Checkmate`	The game is in "Checkmate"
`Stalemate`	The game is in "Stalemate"
Final State Concept	
`WinBlack`	Black player victory
`WinWhite`	White player victory
`Draw`	No player victory

Table 2: List of tokens used in our chess game description language (left) and their corresponding concept (right)

Resource Usage

The resources required to use a new analytical technique are as important as the efficacy of the technique, and affect who benefits from our research (Cook 2021). This was a consideration for us throughout this work. Fortunately, word embedding approaches such as Word2Vec are relatively fast on modest consumer hardware.

We ran our experiments on an Intel i7-7700HQ CPU, using a single thread. Processing the dataset of 250,000 Lichess games into our expanded notation took approximately 1000 seconds, and applying Word2Vec to the dataset to train a word embedding model took approximately 2000 seconds. The whole process took under two hours, and the data preprocessing step only needs to be performed once (should the user want to reuse the data to produce another

Piece	Queen	Rook	King	Bishop	Knight	Pawn
Similarity to "Checkmate"	0.722	0.714	0.612	-0.019	-0.216	-0.943
Relative value	9	5	4	3	3	1

Table 3: The similarity between each piece token and the *Checkmate* token in our model – higher is more similar. Underneath is the piece value assigned by expert players.

Word2Vec model with different parameters).

Using the Model

In the previous section we showed the process of gathering data, processing it into a form amenable to training, and using Word2Vec to construct a model. In this section we show how to use that model to ask and answer questions about the game design and gameplay data. These represent just a few ways in which these models can be used – there are many more exciting applications for this still to be found.

Directly Querying the Model

The simplest way to use the model analytically is to query it directly. We can retrieve the vector representation of any token, but a vector in isolation is of limited use. More commonly, when analysing a word embedding one looks at how vectors relate to one another. The *similarity* of two tokens can be estimated by measuring how far apart they are in vector space using cosine distance. The closer they are to one another, the more similar they are considered to be.

Example 1: Establishing Relative Piece Value As an example of this, we directly queried our model to determine which chess piece is most valuable. This is a common question to ask about many games, whether we want to establish the most powerful cards in a game like Slay The Spire, or are seeking to balance a competitive game like DOTA 2. To establish this, we investigated the similarity of each piece token to the `Checkmate` token. If a piece is strongly related to this token, we hypothesised that this would mean it was more commonly involved in checkmates, and therefore is both more valuable to players and crucial to victory.

Table 3 shows each piece in Chess, along with its similarity to the `Checkmate` token. Recall that a higher value indicates the tokens are more similar to one another. After we retrieved this information from the model, we sought out official rankings of chess piece value used by expert players (Capablanca and de Firmian 2006). This information is shown in the bottom row of the table, and matches the ordering given by our model exactly. We were particularly surprised to see this ordering work even for the King, as most chess masters do not assign a value to the King. We were able to find some orderings, such as Julian Hodgson's, which assign it a value of 4 (Aagaard 2004; Lasker 1988).

Visually Inspecting the Model

As we have explained above, Word2Vec models represent each token in the input language as a vector in a multi-dimensional space. There are many ways to analyse and visualise high-dimensional data, but a simple and common starting point is visualising token similarity. As described in the previous section, token similarity in a word embedding model is measured by how close two vectors are to one another in vector space.

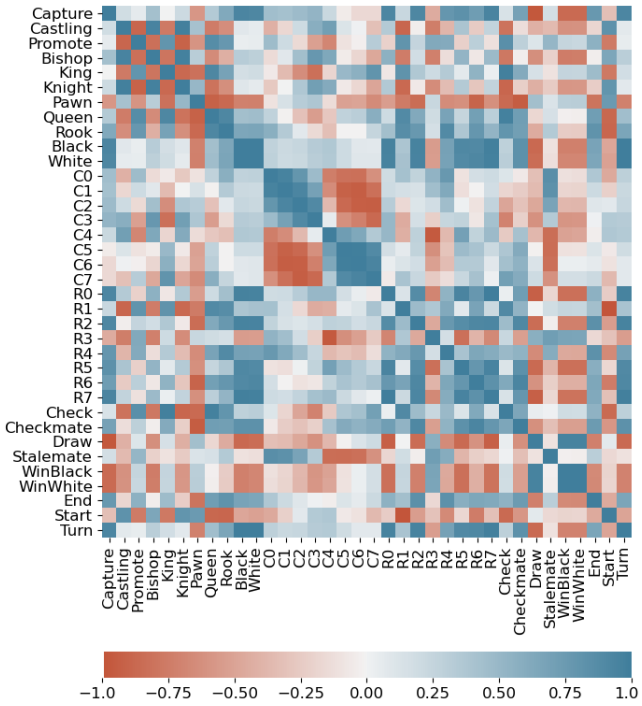

Figure 1: A similarity matrix computed on the 250,000 Lichess dataset from June 2013. Each cell contains a similarity score between two token embeddings.

A simple way to visualise this is to construct a *similarity matrix* which allows us to quickly see the similarity between any pair of tokens in the model. Each row and column of the matrix represents a token, and each cell is colour-coded to represent the cosine distance between the two tokens at that cell, where one colour indicates similarity and another indicates dissimilarity. This process is completely general and can be applied to any Word2Vec model. Figure 1 shows an example similarity matrix calculated for our Lichess dataset.

Similarity matrices make it easier to get a 'birds-eye view' of the model, which can make it easier to make a quick visual assessment of the model. Even with no analysis or insight into the model, we can see from a quick glance at Figure 1 that there are interesting patterns to dive into, with some large blocks of colour, some alternating rows or checkerboard patterns, and some individual points that are strongly coloured surrounded by very white cells. This visual inspection can help us formulate new questions and lead to new queries for the model.

Example 2: Understanding Spatial Relationships Visual representations of the model can help us more easily detect patterns, especially trends between sets of related data. For example, in Figure 1 we can see strong positive and negative relationships between column tokens on either side of the board, reflecting Chess' mirrored nature. We decided to extend our similarity matrix to study the spatial informa-

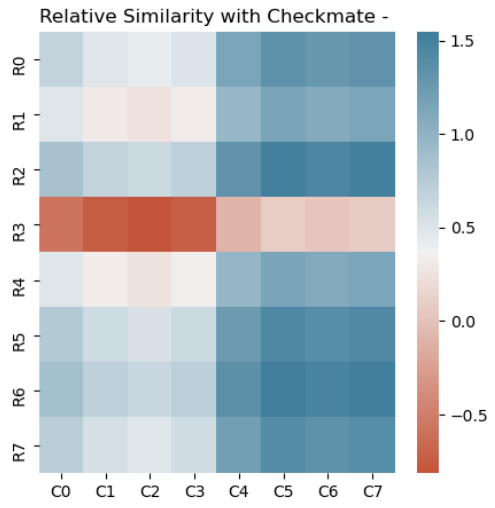

Figure 2: Sum of the similarities with the *Checkmate* vector for each Row vectors (vertical) and Column vectors (horizontal)

tion more closely, by creating a new matrix that combined row and column information relative to a particular token. Specifically, we were interested in how each row and column pairing related to Checkmate, to investigate what parts of the board contribute to this. Our hypothesis was that the result would be broadly evenly distributed.

Figure 2 shows our specialised similarity matrix. Instead of each cell being coloured according to the relative similarity of two tokens, we colour this matrix according to the summed similarity of the row and column with respect to Checkmate. Specifically, for row r_i and column c_j, the heatmap value h_{ij} is defined as:

$$h_{ij} = dist(V(m), V(r_i)) + dist(V(m), V(c_j))$$

Where m is the Checkmate token, $V(x)$ represents the vector of the token x, and $dist$ is the cosine distance function. In other words, a cell (representing a tile on the chessboard) is coloured according to the combined similarity of its row and column with the Checkmate token.

We can see in Figure 2 a clear bias towards the right-hand half of the chessboard. Columns 4-7 generally have a positive similarity with the Checkmate token, while columns 0-3 have a similarity closer to zero (with the exception of Row 3, which replicates this pattern but is shifted closer towards dissimilarity). This result was very surprising to us, as we had anticipated that chess was a largely symmetric game.

However, further research revealed that chess does indeed have a bias towards the right-hand side of the board, as viewed by the White player. We found several data visualisations of chess games, one of a dataset of 400 million games (Reddit User 'Atlas Scrubbed' 2021), another more in-depth analysis of 2 million games (Fırat 2016). In both cases, an evident bias can be seen towards the same side of the board. In his analysis of 2 million games, Fırat also

Piece Name	King	Queen	Rook	Bishop	Knight	Pawn
Similarity to analogy vector	0.983	0.970	0.797	-0.702	-0.819	-0.890

Table 4: A table showing piece similarity for an analogy between promotion (Queen/Pawn) and castling (King/???).

indicates that in 80% of the games he analysed, castling occurred on the king's side (the same side of the board we see checkmate biased towards). This may be one of the key contributing factors to the apparent asymmetry of chess play, as this draws play towards one side of the board.

Complex Direct Queries

In the previous two examples we have focused only on the difference between two vectors in the model's space, to measure similarity. However, there are many other ways to use and analyse word embeddings. One technique that was popularised by Word2Vec models built on English language corpora is the use of vector arithmetic to express relationships between words (Allen and Hospedales 2019). We gave an example of this earlier when we described a process of extracting the capital of Morocco by manipulating vectors representing Paris, France and Morocco: `Paris - France + Morocco` yields a point in space that is closest to the token `Rabat`, the capital of Morocco. The model has captured the 'capital of' relationship, and that can be extracted and reapplied to other tokens.

Example 3: Understanding Special Piece Relationships Gameplay logs typically record player actions, but do not specify why the actions were taken, or what other actions might be possible. For example, castling can only occur using the king and a rook. However, this rule is never made explicit in our data (as illustrated in Table 1), and to an observer with no knowledge of chess there is no particular reason to believe that castling with a bishop would be disallowed. We were curious as to whether the model could infer such special relationships or constraints simply through exposure.

In order to investigate this, we took a similar approach to the analogical approach used in the Paris/France example above. First, we identified another pair of pieces with a special relationship: the pawn and the queen. If a pawn reaches the opposite edge of the board it may be promoted into any other piece type. In our anecdotal survey of chess literature, the chosen piece is almost always a queen. In our notation language this has a special keyword: `Promote`. We attempted to extract the notion of a special link through a key mechanic by performing the following analogy:

`Queen - Pawn + King = ?`

Our expectation here is that the resulting vector from this calculation is the closest to the rook token, since the rook is connected to the king via castling, a similarly special mechanic. Considering only pieces as a possible answer, Table 4 shows a list of pieces and their similarity to the result of this query. Disregarding any results contained within the query (such as Queen) as per (Řehůřek and Sojka 2010), we can see that the top suggestion for this is indeed rook.

This result is notable for two reasons: first, every other candidate answer (Bishop, Knight and Pawn) all show

strongly negative similarity, suggesting the analogical relationship between these pieces is quite strong. Second, although `Castling` is a keyword in the domain language, the model has no concept of what castling involves. There is nothing to suggest a relationship between king and rook other than the raw gameplay data. Thus, this result is a promising suggestion that the model has managed to identify a sense of the connection between these pieces.

Analogies such as this could be used to connect known relationships to speculative ones. For example, we might know that a particular card in *Slay The Spire* is only effective when paired with another card that creates a combo. Using analogical queries such as this, we could extract the notion of 'combo' via a vector subtraction, and then ask the model to suggest synergistic connections between other cards. This is clearly a very complex kind of query, and relies upon fairly nuanced ideas about a game's design that may or may not be deducible from the gameplay logs via a word embedding. We expect further research into this type of query will be needed to establish how widely applicable it is.

Stability of Results

In this section, we investigate how sensible the previous results are to training parameters such as training data size or the random number generator's initial seed. We focus on the one presented in *Example 1*: ranking the tokens associated to each chess piece (`King`, `Queen`, `Rook`, etc) by their similarity to the `Checkmate` token.

Training Data Size To study the impact of training data size on these similarity queries, we created several models with the same parameters and seed, but using different amount of data. (Fig 3)

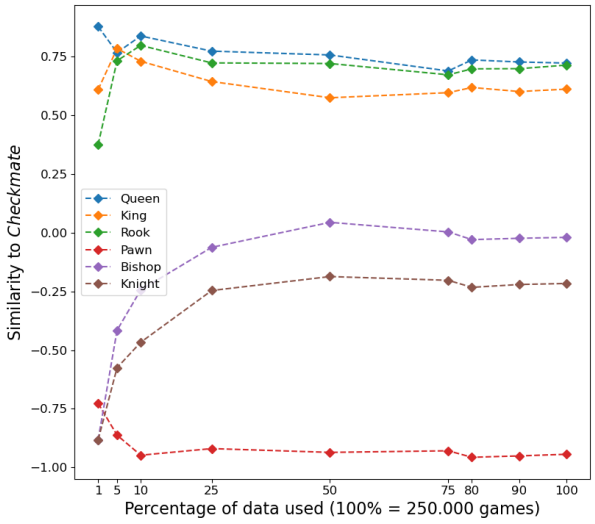

Figure 3: Similarity between *Checkmate* and each chess Piece, in models with various training data size

100% here represents our original dataset of 250,000 chess games. It appears that the interesting ranking result presented in *Example 1* can be observed in models trained with only 10% of the original dataset, and that the exact values for each similarity score between a chess piece token and the `Checkmate` token seem to stabilize around using 25% of the dataset.

These scores suggests that the amount of player logs needed to capture expert chess knowledge within token embeddings might be less than the initial 250,000. Even using around 10,000 game logs (5% of the original dataset) seems to provide a good approximation of the chess piece valuation used by chess experts.

Seed The Word2Vec algorithm uses a random number generator to choose the initial values of each token's vector, as well as to shuffle the order in which the game logs are read (Řehůřek and Sojka 2010).

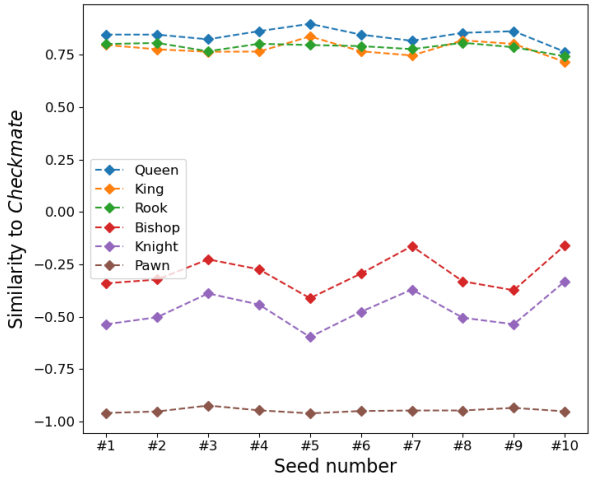

Figure 4: Similarity between *Checkmate* and each chess Piece in 10 models with different seed, each using 10% of the original training data size

In order to study the generator's impact on the similarity queries presented in *Example 1*, we created 10 models using the exact same parameters and dataset, but with different initialisation seeds. Each model uses 10% of the dataset we used to build the original one, as results presented just above showed that this amount was enough for models to capture patterns similar to the ones in the full model.

The results for this experiment are presented in *Figure 4*. It appears that the exact values for each similarity score between `Checkmate` and each piece token are stable across different seeds for the more extreme values (`Queen`, `King`, `Rook`, and `Pawn`) and a bit more widespread for values in the middle (`Bishop`, `Knight`).

On the 10 orderings presented, 7 of them features the exact same piece ranking as the full model, the only difference with the other 3 being that the `King` and `Rook` token are swapped. This is explained by the fact that in this context, the associated similarity scores are very close and small variations can make it so that one token can be considered slightly closer to `Checkmate` than the other one. We expect that averaging a model's representation across several

seeds might mitigate this effect.

Overall, this experiment points towards a satisfactory stability of our model's results across several seeds, minimising the impact of the chance factor in its construction.

Future Work

Use of AI Agents

The experiments we describe above use 250,000 player logs obtained from Lichess. As with many data-driven approaches, more data tends to produce better results. However, most game developers do not easily have access to 250,000 player logs. Data collection is a complex task in itself, which involves solving lots of technical problems, navigating data protection laws, and having a large player base, which is only true for a small percentage of games. Even for large, well-resourced game developers with big playerbases, analysis of this kind are often most useful prior to release, where playtest information is limited and expensive.

AI agents are commonly used to evaluate games in the absence of real players (Hoover et al. 2020). Using AI agents to generate gameplay logs may produce models that are as useful as models generated from player data. Some differences are to be expected – human data will capture a range of skills and emotional states, for example. But we believe that AI agents can still help provide important insights. We have performed some small experiments using game logs generated by the popular Stockfish chess AI engine, and a small dataset of 10,000 game logs yielded a model that allowed us to extract the same information about piece ordering and spatial relationships listed in the previous section.

A larger study is needed to understand the extent to which such data can replace player logs, and the impact of the AI algorithm on the resulting model (for example, a chess engine may already have piece values embedded in it to help it play the game, whereas a more general game-playing agent may not demonstrate the same awareness of piece value). Our hope is that we can conclusively show that AI player logs are a good substitute for real player data, and that perhaps it is possible to combine the two to allow AI agents to support or bootstrap a smaller human dataset. Our initial experiments are promising, and suggest that our approach can be made widely accessible for all game developers.

Use of Expert Data

Our Lichess dataset is not partitioned by skill – it includes players who might be playing their first game of chess, as well as seasoned professionals. It is well understood that for many games experienced players not only play differently, but understand the game in a different way (Lantz et al. 2017). Given that our approach aims to reveal the dynamics of games based on the behaviour of players, we might hypothesise that logs of expert players might reveal different information about the game than logs of novice players. This might allow for the detection and classification of player understanding, skill and playstyle.

We performed an exploratory experiment by building a Word2Vec model using a small dataset of 10,000 tournament games by high-level chess players from the last century. Compared to two similar models made from 10,000-sample slices of the Lichess dataset, the model trained on high-level players is more different from either of the two Lichess slices than the slices are from one another, indicating this approach may be sensitive to player style or skill level. Through a casual initial analysis we noticed, for example, that castling relationships are more strongly expressed in the tournament model, which makes sense as less experienced players are less likely to use this.

We intend to do a larger-scale study in the future, with properly partitioned datasets that group player data by their skill. This could also allow designers to ask separate questions about high- and low-level play, and to see how player behaviour changes as they get better at the game, and could be combined with the use of AI agents to build models trained on data from different AI strengths.

Conclusions

In this paper we have shown how Word2Vec can be used to analyse game logs, revealing detailed and nuanced information about game dynamics. We showed the process of constructing a model, from gathering and preparing data to visualising and analysing the resulting model. We reported on an experiment applying Word2Vec to 250,000 Chess game logs, showing that we were able to rediscover important game knowledge such as the relative value of pieces, and complex, unexpected properties of the game, such as the innate asymmetry of Chess' gameplay.

We believe Word2Vec, when combined with a custom description language for gameplay, is a very promising analytical tool for game developers. Unlike statistical analysis, its aim is not to provide precise measurements of very specific metrics. Instead, this analysis provides a broad overview of the whole game, allowing us to explore it, query it, and find inspiration for further analysis. Our experimentation suggests that this approach can work with light computational budgets, making it accessible to a wide range of users; and that data from AI playouts can provide effective insights too, potentially making it useful even prior to a game's release. In the future, we hope to broaden the use of this technique to provide insight for automated game design systems, as well as exploring other analytical applications for the models.

Acknowledgements

The authors wish to thank the reviewers for their thoughtful feedback which helped improve the final version of this paper. The second author is supported by the Royal Academy of Engineering under the Research Fellowship scheme.

References

Aagaard, J. 2004. *Excelling at technical chess*. London Guilford, CT: Gloucester Distributed in North America by Globe Pequot Press. ISBN 978-1-85744-364-6.

Al-Suli, A. B. 941. *Kitāb al-Shitranj al-Nisha al-Awala*. Quoted in Murray, H. J. R. 1913. *A History of Chess*. Oxford University Press. ISBN 0-19-827403-3.

Allen, C.; and Hospedales, T. M. 2019. Analogies Explained: Towards Understanding Word Embeddings. *CoRR* abs/1901.09813. URL http://arxiv.org/abs/1901.09813.

Browne, C.; and Maire, F. 2010. Evolutionary Game Design. *IEEE Trans. Comput. Intell. AI Games* 2(1): 1–16.

Capablanca, J.; and de Firmian, N. 2006. *Chess Fundamentals*. Random House.

Cook, M. 2021. The Social Responsibility of Game AI. In *Proceedings of the IEEE Conference on Gamees*, 1–8.

Cook, M.; Colton, S.; Gow, J.; and Smith, G. 2019. General Analytical Techniques For Parameter-Based Procedural Content Generators. In *Proceedings of the IEEE Conference on Games*, 1–8.

Fırat, B. 2016. A Visual Look at 2 Million Chess Games. tinyurl.com/ebemunk.

Giovannetti, A. 2019. Slay the Spire: Metrics Driven Design and Balance. Talk at the Game Developers Conference.

Gow, J.; Baumgarten, R.; Cairns, P. A.; Colton, S.; and Miller, P. 2012. Unsupervised Modeling of Player Style With LDA. *IEEE Transactions on Computational Intelligence and AI in Games* 4(3): 152–166.

Hoover, A. K.; Togelius, J.; Lee, S.; and de Mesentier Silva, F. 2020. The Many AI Challenges of Hearthstone. *Künstliche Intelligenz* 34(1): 33–43.

Lantz, F.; Isaksen, A.; Jaffe, A.; Nealen, A.; and Togelius, J. 2017. Depth in Strategic Games. In *The Workshops of the The Thirty-First AAAI Conference on Artificial Intelligence*.

Lasker, E. 1988. *Lasker's chess primer*. London: Portman Press. ISBN 0-7134-6241-8.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]* URL http://arxiv.org/abs/1301.3781. ArXiv: 1301.3781.

Murray, H. J. R. 1986. *A history of chess*. Benjamin Press.

Parrish, A. 2018. Understanding Word Vectors. github.com/aparrish.

Reddit User 'Atlas Scrubbed'. 2021. I looked at a million games played on Lichess and counted how many times checkmate occurred on each square. URL www.reddit.com/r/chess/comments/kp7qwe/i_looked_at_a_million_games_played_on_lichess_and/.

Řehůřek, R.; and Sojka, P. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50. Valletta, Malta: ELRA.

Steven J. Edwards. 1994. *PGN Standard*. Steven J. Edwards. URL http://archive.org/details/pgn-standard-1994-03-12.

Zhong, J.; Nakashima, T.; and Akiyama, H. 2019. A study on the analysis of soccer games using distributed representation of actions and players. ICIC Express Letters.