Transcribing Chess games

Tomson Qin

Australian National University

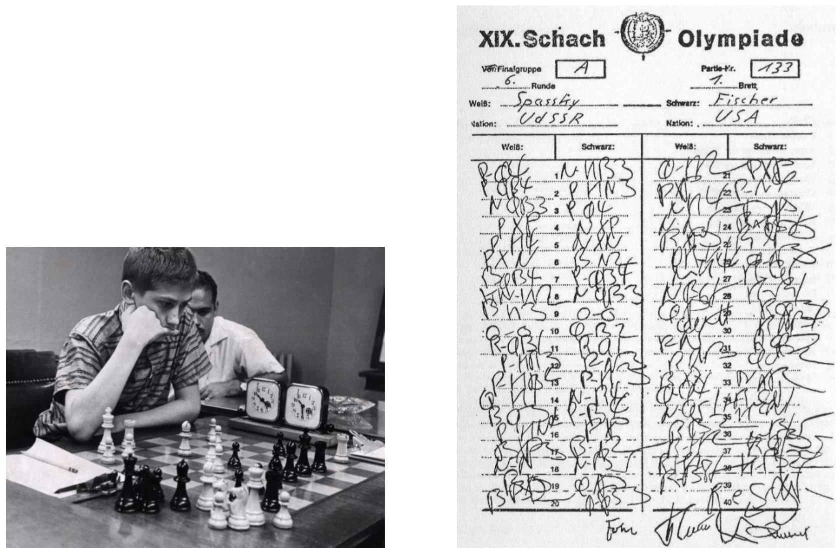
Abstract

A chess player must, upon making a move, record the move on a score sheet like the one in figure 2, according to the rules of the international chess federation. There are many benefits to this formality such as improved notational literacy and a means of accountability, but the main goal is so games can be digitally archived and put on the internet for future reference.

However, just one look at figure 2 and it's clear that the next step to convert handwritten games to digital storage is a daunting one. With tens of thousands of high level chess games being played each week, it is up to volunteers to work through player's score sheets so it can be documented online for the rest of the chess community to study.

I experienced this arduous process first hand in 2013 when I volunteered to document the chess games of the Sydney International Open. The problem arises from illegible handwriting forcing transcribers to look ahead, sometimes tens of moves, to figure out what the likely move was. In a recent conversation with International Chess Arbiter Shaun Press I asked him if the current state of chess documentation was the same as it had been in 2013 and he told me not much has changed at all and that optical character recognition solutions simply "don't work because player's hand writings are too bad".

I think through Bayesian inference, we can determine the missing chess moves based on the moves that we can see. In doing so I wish to provide a fully functional algorithm for the assisted transcribing of chess moves both for humans and OCR processes that makes use of the work done in this project.



(a) A young Bobby Fischer Playing and recording his moves

(b) Bobby Fisher's score sheet from match against Spassky. Reykjavk 1972

Figure 1: Even top players record their own games.

Contents

1. Dataset and programs

1.1. Dataset

The source of the games we look at is TWIC[1], a weekly archive of chess games in PGN format.

1.2. Programs

We make use of a chess position evaluation program called 'stockfish'[2] and a module to interact with chess moves called [3].

2. Single Missing Move

2.1. An example

Move	White	Black
1	e4	e5
2	f4	exf4
3	Nf3	Be7
4	Bc4	??
5	Kf1	d6
6	d4	Nc6

Table 1: Black's 4th move is missing

As we can see, black's 4th move is either missing or too illegible to make out, how should we proceed?

A naive approach to solving this problem is to continue on the position from

Figure 2: White's 5th move likely caused by black's bishop checking on h4.

every possible choice of black's 4th move and prune away the moves that eventually lead to an illegal move (a move that cannot be played). However, this quickly becomes computationally infeasible if there are multiple unreadable notations in a small number of moves. Furthermore, the game could simply end before the possibilities of black's 4th move is narrowed down to one possibility.

Better is to use the likelihood of future moves conditional of black's 4th move. Without getting into depth about what's actually happening on the chessboard, it suffices to know that White's 5th move is not something you would play willingly. We can infer black's 4th move is very likely to have been the move 4..Bh4+ which forces white to play 5.Kf1 or be put in a very dangerous position. In other words, white's 5th move is the worst move in every possible position except the one where black plays 4..Bh4+ in which case it is one of the best moves.

What we have essentially done is use Bayes theorem to infer black's 4th move based on the likelihoods of the subsequent moves.

2.2. The Probability of picking a move

Now we need to develop a formal method to evaluate the probability a player chooses a move given a position. To do this, we make use of "chess engines" which are algorithms that give evaluations for positions in centipawn values. The metric used is called centipawn meaning one hundredth value of a pawn. Being up 100 centipawns is equivalent to being up a pawn, which is an advantage easily interpretable by chess players. Roughly speaking, being up 100 centipawn is a nice advantage but is usually hard to convert to a win. On the other hand, being up 300 centipawns or more should be decisively winning, although depending on the complexity of the position may not be obvious for human players.

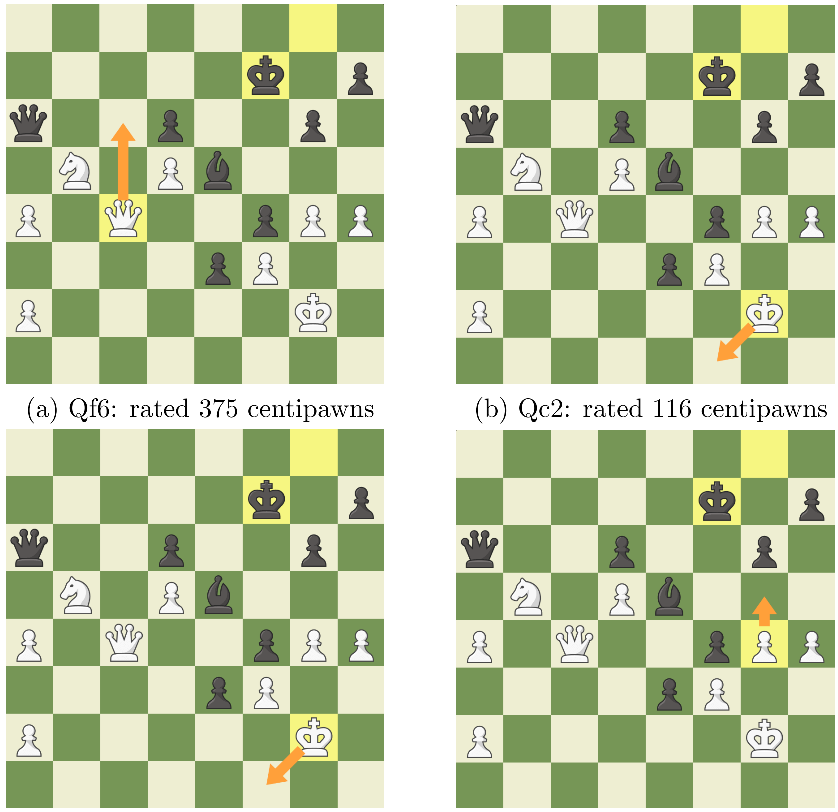
Figure 3: White to play

In table 2 white to play has a variety of moves to choose from. However not all moves are created equal and the Engine gives different evaluations to each move as shown in figure 5.

Move	**White**	**Black**
39	h4	Kf7
40	?? (Qb4)	h5
41	g5	Kg8
42	Qc4	...

Table 2: White's 40th move is missing.

In this position, Qf6 stands out as the best move giving an evaluation



(a) Qf6: rated 375 centipawns

(b) Qc2: rated 116 centipawns

(c) Kf1: rated 104 centipawns

(d) g5: rated 86 centipawns

Figure 4: Evaluations of best continuations

of 375 centipawns, a significant advantage. The next best alternative is Qc2 with an evaluation of 116 centipawns. Finally, there are also losing moves, expressed with negative centipawn values, which give black the advantage.

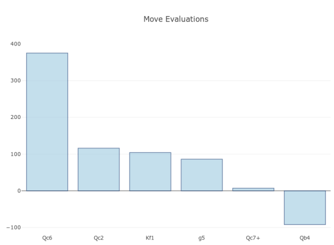
To weaker chess players what to do in this position is not clear. The engine may say that Qc6 is the best move in the position and whilst it does looks strong I could still see myself overlooking it. However, that is not to say that weaker players actively try to avoid good moves. It is more a problem of there being too many moves and most of them weak. If a player actively looks for good moves I imagine the probability of each move being selected contains a deterministic component based to the underlying strength of the move and the stochastic component based on the player's skill. The better a player is, the more the deterministic merit of the move shine through the noise of a player's decision making. More on this in the last section.

Here is a plot (figure 6) of a few moves and the the centipawn evaluations after making them. Qb4 gives negative centipawn evaluations because after making it, the opposing side will hold the advantage.

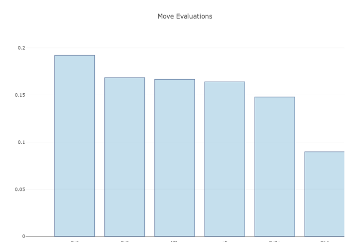
If these values weren't negative, things begin to look like a sensible discrete probability distribution function. So we apply the following function to the centipawn evaluation of every candidate move:

$$P(A|Position) = (E(A|Position) + c)^k$$

where A is a potential move, E is the engine evaluation, $k \epsilon [0, 1]$ the uncertainty and c some value that scales with the value of the minimum evaluation to make the centipawn evaluations of every proposed move be positive and reasonably scaled relative to one another. The choice of k reflects the uncertainty of human choice from machine recommendations. Finally, we normalize over all values to get a distribution function (figure 7).



(a) Engine's centipawn evaluation of moves



(b) After transforming evaluations to probability distribution

Figure 5: Evaluations of best continuations

Now that we can assign probabilities to moves, we can now use bayes theorem to find the probability of missing moves. The process is illustrated in the appendix code, but in summary the probability for a chain of moves is the running product of probabilities of future moves given the position arrived at through playing the chain of moves. We only normalize over all probabilities at the end of the chain. Naturally, any assumption of move 40 which leads to an illegal move will have probability 0 of being the true move 40.

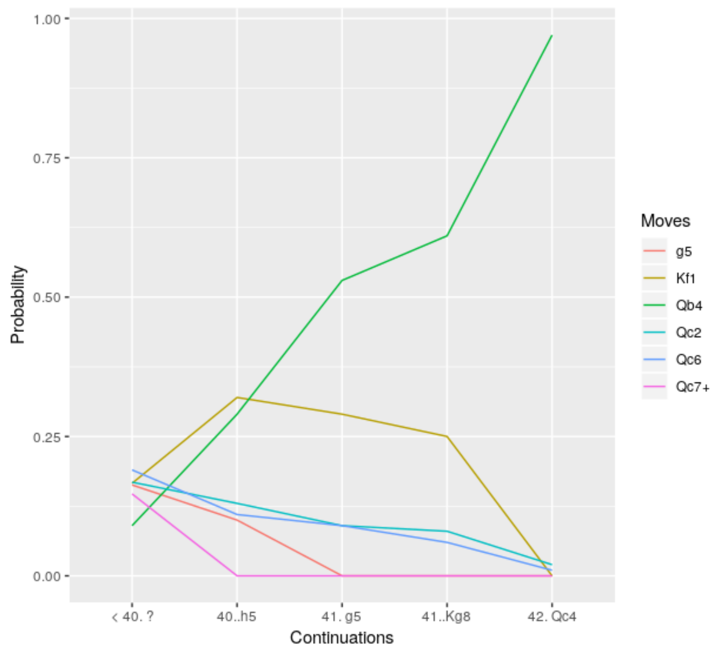


Figure 6: Probability of candidate moves given future moves

2.3. Results

Given our 5 candidate moves, we can see the probability of each continuation change with more and more future move information. After 4 moves, 3 candidate moves are relegated to being impossible, 2 highly unlikely and 1 move clearly most likely. We see that the most likely Qb4 was indeed the move played.

3. Multiple Missing moves

3.1. Another Example

Our model can be generalized to determining the likelihood of multiple missing moves by applying branching at each additional missing move.

Since the due date of this project was postponed, I decided to change the following example to the first match of the 2018 world chess championships played on the 9th of November.

After black's 19th move the position reached Figure 9.

The continuation from black's 19th move with every second white move missing is shown in table 4.

Figure 7: The position after move 19. Caruana V Carlsen. Holborn 2018

Move	White	Black
5	d3	Bg7
6	h3	Nf6
7	Nc3	Nd7
8	Be3	e5
9	0-0	b6
10	Nh2	Nf8

Table 3: Round 1 of Carlsen v Caruana

Move	White	Black
5	?	?
6	h3	Nf6
7	Nc3	Nd7
8	Be3	e5
9	0-0	b6
10	Nh2	Nf8

Table 4: 2 moves are treated as missing

3.2. Method

In this model, I iterated through legal moves for white's 5th move then picked the top 10 moves according to the evaluation and repeated to find black's top 10 moves conditional on every one of white's 5th move. In all, this creates $10 \times 10 = 100$ possible combinations of possible 5th move. Then I calculated the probability of the game continuing on as it did in the actual game. Figure 10's plot shows the updating of the probability of each combination of move 5 given more and more information about how future play went.

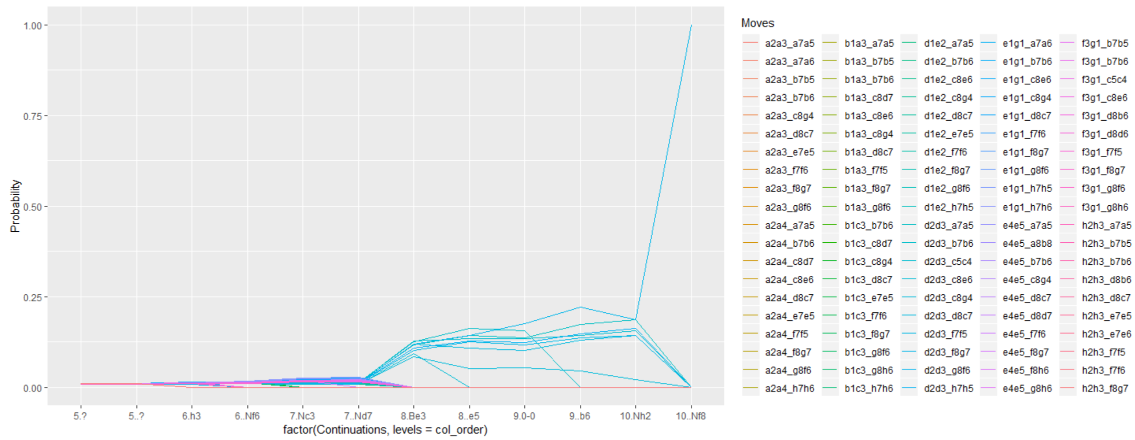


Figure 8: Continuations from move 4. Caruana V Carlsen. Holborn 2018

3.3. Results

After 2 moves by each player blue lines dominate our plot, this tells us we've already pinpointed white's 5th move. After another move by each player, black and white's true 5th move is already ranked most likely. In another 2 moves by each player, we know white and black's 5th move for certain conditional on white and black moves being amidst the original candidate moves.

This is significant for several reasons. Firstly, If the game ended at move 9 because a player forfeits, we would still have reasonable predictions for the 5th move. Secondly, this probabilistic approach lets us prune highly unlikely moves from consideration, thereby letting us look at more candidate moves than we would otherwise be able to. Thirdly, the fact that our algorithm showed good results here in the opening, where no move is likely to win/lose you the game was very unexpected, this method would no doubt work even better for middle games and end games where only a few moves are viable.

4. Probability of failure

One of the main problems to this approach is when human play falls largely outside machine's top moves. If a player plays a move so bad that it falls outside the top 20 moves in a position, our prediction will inevitably be wrong. It would be good if we could estimate the probability of such an error occurring beforehand and set the number of candidate moves as to keep the probability of failure under some $\epsilon\%$. Let us make use of the fact that some people are simply more prone to making errors than others.

A good measure of the probability of player's true move falling outside the number of candidate moves is the probability of a blunder. I went through 8 of Fibianno Caruana's games and calculated the computer evaluation before and after each of his moves and recorded the change in centipawn evaluation. If the evaluation differed by more than 100 centipawns, it means that caruana has likely made a blunder. I recorded the number of moves I went through (exposure) and the number of blunders made. I repeated this for several other players and built a hierarchical model, using the code from assignment 4, with blunder rates nested within players. With shrinkage, caruana has about a 2% chance of making a 100 centipawn blunder. The point is, through shrinkage, I can have reasonable estimates of blunder rates even for new players with one recorded game.

The higher the rate of blunder, the more candidate should employed. Furthermore, the transformation from centipawn evaluations to probability of playing each move should also be some function of blunder rate. Specifically, weaker players might have a flatter move choice distribution than stronger players as they are less able to distinguish the merits of moves.

5. Discussion

5.1. Improvements on OCR

This method can be easily combined with existing OCR framework. As can be seen in figure 9, the accuracy of existing OCR is pretty awful. Perhaps a multivariate distribution where you look at both the likelihood of the moves and the similarity of those move's notation to the observed handwriting can be used.

5.2. The problem of consecutive bad entries

If one side had multiple illegible entries in close succession and those moves could be interchanged, then further information in the form of contin-

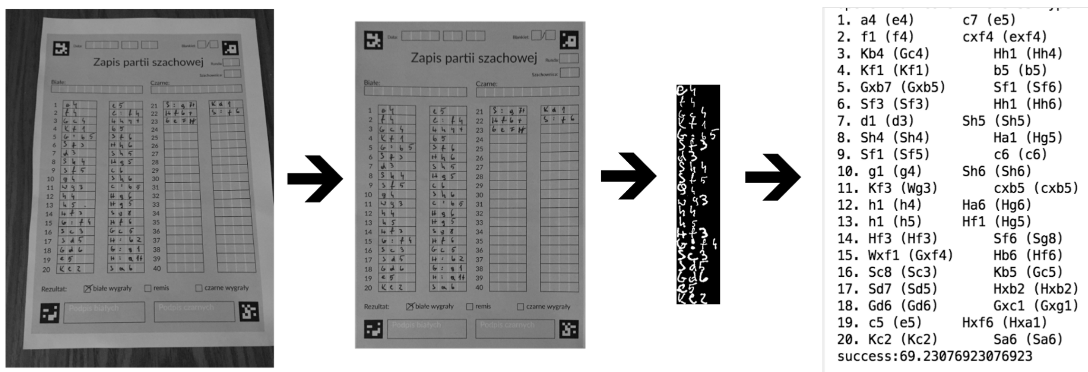
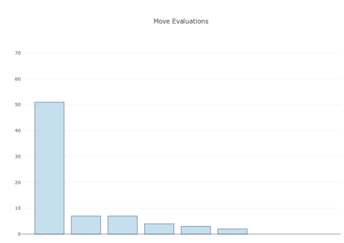


Figure 9: An example of current OCR technology in chess [4]. The author only does a check to see if proposed moves are legal not the likelihood of moves as we have done.

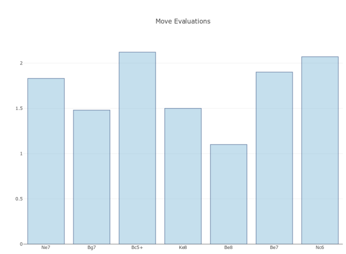
uations will only show you what those moves were but won't shed any more light on which order you played those two moves.

5.3. Machine Evaluation as basis for probability

The use of machine evaluation to give probabilities to different moves by human players isn't very accurate especially when the position isn't at a crucial stage. In figure 10 I plotted the frequency of moves picked by human players and the evaluation of machines for a position in the opening.



(a) Histogram of player moves



(b) Machine evaluation of moves

Figure 10: Evaluations of best continuations

As you can see, they look quite different. There are two likely reasons for this. Firstly, in order to be competitive, chess players study and memories the same reputable openings which usually grows with popularity. Secondly, humans generally use reoccurring themes and tactics such as the fork, the pin and the skewers as a heuristic to see which moves they should explore, causing a bias in move selection.

Nonetheless, there were about 50 moves black could have played in this position and if you rank them in terms of evaluations, all of the moves the players chose appear in the top 12. This is sufficient reason to use this method to estimate the probability of players especially after seeing the results we have shown.

6. Conclusion

In conclusion, we explored the use of bayesian score sheet imputation methods that were demonstrated to work for opening and middle games positions. With multiple missing moves, our method is able to focus on only the most likely lines and isolate true moves among one hundred possible choices when given an additional 5 moves. Then we used a Bayesian Hierarchical model to estimate the likelihood of failure and assign parameters accordingly. Lastly, we discussed some of the improvements that can be made to existing OCR techniques as well as some of the theoretical problems of using computer recommendations to estimate probability of playing moves. In all, Bayes theorem is perfectly suited to the problem of missing move estimation and should have been applied for the purpose ages ago. The code in the appendix is already functional and can be used for any position, but it doesn't do any pruning yet so takes a long time when there's multiple missing moves.

7. Bibliography

[1] M. Crowther, This week in chess games archive, 2018.

[2] D. Yang, Stockfish, 2018.

[3] Niklasf, python-chess, https://github.com/niklasf/python-chess/blob/master/docs/core.rst, 2018.

[4] M. migielski, From chess score sheet to icr, 2018.

8. Appendix