Appendix

A Acknowledgement

We thank Jiacheng Liu for his work on collecting chess-related data and chess book list.

B Different Chess Formats

B.1 Universal Chess Interface (UCI)

The UCI format is widely used for communication between chess engines and user interfaces. It represents chess moves by combining the starting and ending squares of a piece, such as "e2e4" to indicate moving the pawn from e2 to e4. For example, the UCI notation for a full game would be:

```
e2e4 c7c6 g1f3 d7d5 e4d5 c6d5 d2d4 b8c6 c2c4 g8f6 b1c3 c8e6 c4c5 g7g6 c1f4 f8g7
f1e2 f6e4 e1g1 e6g4 f3e5 g4e2 d1e2 c6e5 c3e4 e5c6 e4d6 e8f8 f1e1 g7d4 f4h6 d4g7
h6g7 f8g7 d6b5 a7a6 b5c3 d5d4 c3e4 d8d5 a2a3 a8d8 b2b4 h7h6 e2
```

B.2 Standard Algebraic Notation (SAN)

SAN (Standard Algebraic Notation) is a widely used notation system in the game of chess for recording and describing moves. It provides a standardized and concise representation of moves that is easily understood by chess players and enthusiasts. In SAN, each move is represented by two components: the piece abbreviation and the destination square. The piece abbreviation is a letter that represents the type of piece making the move, such as "K" for king, "Q" for queen, "R" for rook, "B" for bishop, "N" for knight, and no abbreviation for pawns. The destination square is denoted by a combination of a letter (a-h) representing the column and a number (1-8) representing the row on the chessboard. Additional symbols may be used to indicate specific move types. The symbol "+" is used to indicate a check, while "#" denotes a checkmate. Castling moves are represented by "O-O" for kingside castling and "O-O-O" for queenside castling.

B.3 Portable Game Notation (PGN)

PGN is a widely adopted format for recording chess games. It includes not only the SAN moves but also additional information like player names, event details, and game results. PGN files are human-readable and can be easily shared and analyzed. Here is an example of a PGN representation of a full game:

```
[Event "World Chess Championship"]
[Site "London, England"]
[Date "2023.05.20"]
[Round "1"]
[White "Carlsen, Magnus"]
[Black "Nepomniachtchi, Ian"]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6
8. c3 O-O 9. h3 Nb8 10. d4 Nbd7 11. Nbd2 Bb7 12. Bc2 Re8 13. Nf1 Bf8
14. Ng3 g6 15. a4 c5 16. d5 c4 17. Bg5 h6 18. Be3 Nc5 19. Qd2 h5 20. Bg5 Bg7
21. Nh2 Qc7 22. Rf1 Nh7 23. Bh6 Bh8 24. f4 exf4 25. Bxf4 Nf6 26. Rae1 bxa4
27. Nf3 Nfd7 28. Bh6 Ne5 29. Nxe5 Bxe5 30. Rf3 Qb6 31. Kh1 Qxb2 32. Ref1 Re7
33. Bg5 Rd7 34. Bf6 Bxf6 35. Rxf6 a3 36. Nxh5 a2 37. Qh6 gxh5 38. R6f3 h4
39. Rf4 f5 40. Rxf5 Rg7 41. Rh5 1-0
```

B.4 Forsyth–Edwards Notation (FEN)

FEN is a notation system used to describe the state of a chess game. It represents the positions of pieces on the chessboard, active color, castling rights, en passant targets, and the half-move and full-move counters. Here is an example of a FEN notation representing the starting position:

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
```

In this FEN notation, each letter represents a piece on the board, with uppercase letters representing white pieces and lowercase letters representing black pieces. The forward slash ("/") separates ranks, and the number after each rank indicates the number of empty squares. The active color is represented by "w" for white or "b" for black. The castling rights are denoted by "K" (white kingside), "Q" (white queenside), "k" (black kingside), and "q" (black queenside). The en passant target square is indicated with the corresponding square, or "-" if there is no en passant target. The half-move and full-move counters specify the number of half-moves since the last pawn move or capture and the current move number, respectively.

These different chess formats serve various purposes, from representing individual moves (UCI) to recording entire games (PGN) and describing specific positions (FEN). Understanding and working with these formats is essential for tasks like parsing, analyzing, and exchanging chess game data in different contexts.

C Image sources

Here we provide the sources for images used in our Figure 1.

- Online game: https://lichess.org/tv
- Pro game: https://www.insidethegames.biz/articles/1110959/magnus-carlsen-chess-world-cup-duda
- Computer game: https://www.chess.com/news/view/updated-alphazero-crushes-stockfish-in-new-1-000-game-match
- Modeling: https://www.chess.com/terms/fen-chess
- Puzzle: https://chesspuzzle.net/
- General text: https://www.bbc.co.uk/newsround/66233770
- Wiki: https://en.wikipedia.org/wiki/Chess
- Forum: https://www.chess.com/forum
- Blog: https://www.chess.com/blogs
- Book: https://www.amazon.co.uk/Chess-Strategy-Action-John-Watson/dp/1901983692
- Annotated game: https://www.chess.com/forum/view/game-analysis/kingside-pawn-rush
- Youtube video: https://www.youtube.com/watch?v=NAIQyoPcjNM
- Reddit: https://www.reddit.com/r/chess/comments/15s65hb/white_to_play_and_reach_2000_chesscom/jwdqpze/?context=3
- Value judgement: https://www.thenewatlantis.com/wp-content/uploads/legacy/20190820_TNA58Wilkenfeldbanner.jpg

D Dataset analysis

D.1 Dataset statistics and metrics

InTable 9, we present the dataset statistics breakdown for each data subset, including its raw size, document count, and subset type.

Table 10 shows the properties of the chess-specific language dataset that we use for training ChessGPT. For these datasets, we computed the average number of words (num. words) per example, character repetition ratio (char. rep. ratio), word repetition ratio (word. rep. ratio.), special character ratio (special char. ratio), stopwords ratio and perplexity of the first shard for each subset in the language dataset. These metrics are based on some of the criteria employed by Bloom [25] in their pre-processing pipelines.

Table 9: Dataset statistics

Component	Raw size	Document count	Subset type
Lichess	19.9 GB	17.5 M	Game
Pro-player	0.37 GB	0.44 M	Game
CCRL	3.60 GB	3.00 M	Game
Chess puzzles	1.53 GB	3.19 M	Game
Chess modeling	0.94 GB	1.90 M	Game
C4-Chess	0.59 GB	0.18 M	Language
Pile-Chess	1.10 GB	0.10 M	Language
RedPajama-Chess	5.65 GB	0.52 M	Language
Oscar-Chess	3.13 GB	0.33 M	Language
WikiPedia-Chess	40.3 MB	11.4 K	Language
Chess Blogs	0.59 GB	73.2 K	Language
Chess Books	1.86 GB	8.36 K	Language
Chess Forums	1.05 GB	0.14 M	Language
Annotated Chess Games	0.66 GB	245 K	Mixed
Youtube transcripts	0.98 GB	83.0K	Mixed
GPT-4-Chess	0.95 MB	3.91 K	Conversation
Reddit	0.86 GB	0.41 M	Conversation
Overall	42.3 GB	28.1 M	N/A

Table 10: Metrics for the language dataset.

dataset	num. words	char. rep. ratio	word rep. ratio	special char ratio	stopwords ratio	perplexity
Chess puzzles	49.6618	0.0104	0.0000	0.3246	0.4771	988.3734
Oscar-Chess	1441.4341	0.0596	0.0499	0.2242	0.4119	665.6499
Pile-Chess	2105.2454	0.0626	0.0205	0.2409	0.4227	497.3883
RedPajama-Chess	1581.5825	0.0532	0.0163	0.2273	0.4218	410.1236
StackExchange-Chess	578.3733	0.0591	0.0816	0.2617	0.4835	520.1257
Wikipedia-Chess	463.4980	0.0876	0.0052	0.2604	0.3079	236.9671
C4-Chess	510.6041	0.0479	0.0082	0.2131	0.4418	548.9471

D.2 Dataset examples

In this subsection, we offer dataset examples for each subset to help the readers to understand its contents more intuitively and clearly.

Lichess

```
[Event "Rated Bullet tournament https://lichess.org/tournament/yc1WW2Ox"]
[Site "https://lichess.org/PpwPOZMq"]
[Date "2017.04.01"]
[Round "-"]
[White "Abbot"]
[Black "Costello"]
[Result "0-1"]
[UTCDate "2017.04.01"]
[UTCTime "11:32:01"]
[WhiteElo "2100"]
[BlackElo "2000"]
[WhiteRatingDiff "-4"]
[BlackRatingDiff "+1"]
[WhiteTitle "FM"]
[ECO "B30"]
[Opening "Sicilian Defense: Old Sicilian"]
[TimeControl "300+0"]
[Termination "Time forfeit"]
1. e4 { [%eval 0.17] } 1... c5 { [%eval 0.19] }
2. Nf3 { [%eval 0.25] } 2... Nc6 { [%eval 0.33] }
3. Bc4 { [%eval -0.13] } 3... e6 { [%eval -0.04]  }
4. c3 { [%eval -0.4]  } 4... b5? { [%eval 1.18]  }
5. Bb3?! { [%eval 0.21]  } 5... c4 { [%eval 0.32]  }
6. Bc2 { [%eval 0.2]  } 6... a5 { [%eval 0.6]  }
7. d4 { [%eval 0.29]  } 7... cxd3 { [%eval 0.6]  }
8. Qxd3 { [%eval 0.12]  } 8... Nf6 { [%eval 0.52]  }
9. e5 { [%eval 0.39]  } 9... Nd5 { [%eval 0.45]  }
10. Bg5?! { [%eval -0.44]  } 10... Qc7 { [%eval -0.12]  }
11. Nbd2?? { [%eval -3.15]  } 11... h6 { [%eval -2.99]  }
12. Bh4 { [%eval -3.0]  } 12... Ba6? { [%eval -0.12]  }
13. b3?? { [%eval -4.14]  } 13... Nf4? { [%eval -2.73]  } 0-1
```

Pro-player

```
[Event "URS-ch34"]
[Site "Tbilisi"]
[Date "1966.??.??"]
[Round "9"]
[White "Bronstein, David I"]
[Black "Suetin, Alexey S"]
[Result "1/2-1/2"]
[WhiteElo ""]
[BlackElo ""]
[ECO "B97"]

1.e4 c5 2.Nf3 d6 3.d4 cxd4 4.Nxd4 Nf6 5.Nc3 a6
6.Bg5 e6 7.f4 Qb6 8.Qd2 Qxb2
9.Rb1 Qa3 10.Bxf6 gxf6 11.Be2 Bg7 12.O-O Nc6
13.Nxc6 bxc6 14.Rb3 Qc5+ 15.Kh1 f5
16.exf5 exf5 17.Na4 Qd4 18.Qxd4 Bxd4 19.Rd1 Bf2
20.Rxd6 O-O 21.Nb6 Bxb6 22.Rxb6 Re8
23.Bf1 Be6 24.Kg1 Bxa2 25.Rxa6 Rxa6 26.Bxa6 Bd5
27.Kf2 Re4 28.g3 Bc4 29.Rxc6 Re2+
30.Kg1 Bxa6 31.Rxa6 Rxc2 32.Ra5 Kg7 33.Rxf5 Kg6
34.Rg5+ Kf6  1/2-1/2
```

CCRL

```
[Event "CCRL 40/15"]
[Site "CCRL"]
[Date "2022.01.08"]
[Round "806.6.381"]
[White "Stockfish 060122 64-bit"]
[Black "Dragon by Komodo 2.6 64-bit"]
[Result "1/2-1/2"]
[ECO "D30"]
[Opening "Queen's gambit declined"]
[PlyCount "115"]
[WhiteElo "3505"]
[BlackElo "3480"]

1. d4 {book} d5 {book} 2. c4 {book} e6 {book} 3. Nf3 {book} Nf6 {book}
4. g3 {book} a6 {book} 5. c5 {book} b6 {book} 6. cxb6 {+0.23/33 28s}
c5 {-0.23/30 40s} 7. Bg2 {+0.24/30 11s} cxd4 {-0.15/29 17s} 8. Nxd4
{+0.08/32 15s} Nbd7 {-0.21/29 18s} 9. Nc3 {+0.29/30 18s} Nxb6
{-0.31/29 21s} 10. O-O {+0.27/30 12s} Bb7 {-0.21/28 20s} 11. Be3
{+0.41/31 14s} Nc4 {-0.12/27 19s} 12. Bg5 {+0.31/30 18s} Be7 {-0.16/29
29s} 13. Qa4+ {+0.24/29 18s} Qd7 {-0.08/30 20s} 14. b3 {+0.07/33 22s}
Nd6 {+0.00/29 17s} 15. Rfc1 {+0.19/31 28s} Rc8 {-0.02/31 15s} 16. e3
{+0.13/33 24s} Nde4 {-0.02/33 16s} 17. Qxd7+ {+0.27/34 21s} Kxd7
{-0.05/34 31s} 18. Nxe4 {+0.25/37 13s} Nxe4 {-0.17/32 22s} 19. Bxe7
{+0.29/32 22s} Kxe7 {-0.24/32 20s} 20. Bxe4 {+0.19/35 36s} dxe4
{-0.12/34 37s} 21. Kf1 {+0.13/38 88s} g5 {-0.03/33 27s} 22. Ne2
{+0.00/37 63s} Kd6 {-0.03/36 21s} 23. Rxc8 {+0.08/40 12s} Rxc8
{-0.08/37 25s} 24. Ke1 {+0.12/39 20s} Bd5 {-0.05/34 40s} 25. Rc1
{+0.07/41 17s} Ra8 {-0.06/32 27s} 26. Kd2 {+0.05/33 16s} a5 {+0.00/35
32s} 27. Nc3 {+0.10/35 18s} Rb8 {-0.05/37 20s} 28. Rh1 {+0.04/39 16s}
Ke5 {+0.00/39 67s} 29. Rc1 {+0.07/43 56s} f5 {+0.00/39 21s} 30. Na4
{+0.00/41 13s} Kd6 {+0.00/40 19s} 31. Rc5 {+0.00/46 34s} Ra8 {+0.00/42
57s} 32. Rc2 {+0.00/49 20s} Rb8 {+0.00/40 13s} 33. Nb2 {+0.12/37 17s}
h5 {+0.00/41 14s} 34. Nc4+ {+0.11/41 29s} Bxc4 {+0.00/42 14s} 35. Rxc4
{+0.00/46 23s} h4 {+0.00/43 21s} 36. Ra4 {+0.00/48 20s} Rb5 {+0.00/43
16s} 37. Kc3 {+0.00/50 29s} Ke5 {+0.00/44 18s} 38. b4 {+0.00/51 24s}
axb4+ {+0.00/41 26s} 39. Rxb4 {+0.00/53 30s} Rc5+ {+0.00/43 14s} 40.
Kb3 {+0.00/53 26s} Rd5 {+0.00/43 20s} 41. a4 {+0.00/50 20s} Kd6
{+0.00/45 31s} 42. Rc4 {+0.00/50 16s} h3 {+0.00/48 25s} 43. Kc2
{+0.00/53 34s} Kd7 {+0.00/47 23s} 44. Kb3 {+0.00/47 17s} Rd3+
{+0.00/49 22s} 45. Kc2 {+0.00/50 19s} Rd5 {+0.00/51 28s} 46. Rc3
{+0.00/52 35s} g4 {+0.00/53 22s} 47. Kc1 {+0.00/57 23s} Ra5 {+0.00/55
25s} 48. Rc4 {+0.00/58 21s} Ra6 {+0.00/57 21s} 49. Kc2 {+0.00/59 59s}
Rd6 {+0.00/57 28s} 50. a5 {+0.00/58 14s} Rd5 {+0.00/53 23s} 51. Ra4
{+0.00/60 27s} Kc7 {+0.00/57 16s} 52. a6 {+0.00/60 17s} Kb8 {+0.00/60
18s} 53. a7+ {+0.00/59 25s} Ka8 {+0.00/65 42s} 54. Ra6 {+0.00/59 20s}
Rc5+ {+0.00/66 16s} 55. Kb2 {+0.00/65 25s} Rd5 {+0.00/66 14s} 56. Kc2
{+0.00/62 17s} Rc5+ {+0.00/66 18s} 57. Kd1 {+0.00/67 17s} Rd5+
{+0.00/63 14s} 58. Kc2 {+0.00/101 20s, Draw by 3-fold repetition}
1/2-1/2
```

Chess puzzles

```
Try your hand at this chess puzzle. The board's FEN is
1r4k1/4nppp/8/4Pb2/8/1P5P/r1PR4/3R3K w - - 0 27, and you need to
determine the optimal move for the player. This puzzle focuses on
backRankMate,endgame,mate,mateIn2,short, and the solutions are
provided in both SAN format as d2d8,b8d8,d1d8 and UCI format as 27.
Rd8+ Rxd8 28. Rxd8#.
```

Chess modeling

```
With the FEN board state 3r4/4Rpp1/2N2k1p/8/5B2/5BP1/P4PKP/8 b - - 6
42 and a move in UCI d8d6, what is the corresponding SAN move? The
derived move is Rd6.
```

C4

```
Nothing improves your chess more than playing long time control
tournament events. The Irish championships are being held in Dublin
next weekend. There are a lot of events to choose from! We have a
message encouraging us to play from the Irish Chess Union Chairman.
All members of our club are registered (by the club) with the ICU. You
are eligible to play and should consider playing!
```

Pile

```
Q: What is the theory behind center control? Center control is an
important aspect of playing chess, most openings are built around
controlling the center, but why? Is center control really that
important for winning a game?
```

RedPajama

```
... Mark Dvoretsky - strong player and fantastic coach Mark
Israilewitsch Dvoretsky was born on 9th December 1947 in Moscow. After
finishing his studies of Mathematics and Economics in 1972 Dvoretsky
focused on a career as chess trainer and among other things worked for
Botvinnik's school of chess. As a young player Dvoretsky achieved a
number of notable sucesses: in 1973 he won the Championship of Moscow
and in 1974 he finished fifth at the USSR-Championship in Leningrad.
One year later, in 1975, he won the B-tournament in Wijk aan Zee. But
he soon decided to focus on his career as a chess trainer Dvoretsky
has trained countless strong players, and among his regular students
are well-known players such as Valery Chechov, Nana Alexandria, Sergei
Dolmatov, Alexej Dreev and Artur Jussupow. Among the players who
occasionally trained with Dvoretsky are Garry Kasparov, Viswanthan
Anand, Veselin Topalov, Evgeny Bareev, Viktor Bologan, Loek van Wely
and lots of others. One training method of Dvoretsky was to play
selected positions with both colors against his students - and he
often surprised his students by winning the same position with Black
and with White. Dvoretsky was an International Master and FIDE Senior
Trainer. He published a number of textbooks, sometimes with Artur
Jussupow as co-author. ChessBase published a digital version of his
"Endgame Manual". Dvoretsky was a firm part of chess life in Moscow
and popular guest at chess events all over the world. Russian Chess
Federation ...
```

Oscar

```
If one wishes to learn chess from some of the greatest players in the
world, but does not live in greater New York, then online lessons may
be what he or she is looking for. Many of our coaches are experienced
in teaching both group and solo online lessons. Online lessons are
orchestrated via Skype using an online chess program.
```

Chess Blogs

```
What Is The Elo Rating System? The Elo rating system measures the relative
strength of a player in some games, such as chess, compared to other players.
Its creator, Arpad Elo, was a physics professor in the United States and a chess
master who worked to improve the way the U.S. Chess Federation measured their
players' skill levels. He was a solid chess player himself, as you can see from
this game he played against a young Bobby Fischer.

[Event "New Western Open"]
[Site "Milwaukee, WI USA"]
[Date "1957.07.04"]
[Round "2"]
[White "Arpad Elo"]
[Black "Robert James Fischer"]
[Result "0-1"]
[EventDate "?"]
[ECO "B93"]
[WhiteElo "?"]
[BlackElo "?"]
[PlyCount "98"]

1. e4 c5 2. Nf3 d6 3. d4 cxd4 4. Nxd4 Nf6 5. Nc3 a6 6. f4 e5 7. Nf3 Qc7 8. Bd3
Nbd7 9. O-O b5 10. Qe1 Bb7 11. a3 g6 12. Qh4 Bg7 13. g4 exf4 14. Bxf4 O-O 15.
Qg3 Ne5 16. Nxe5 dxe5 17. Bxe5 Qc5+ 18. Rf2 Nh5 19. Bd6 Qxc3 20. bxc3 Nxg3 21.
Bxf8 Rxf8 22. hxg3 Bxc3 23. Rb1 Bd4 24. a4 Bc8 25. axb5 axb5 26. Rxb5 Bxg4 27.
Kg2 Bxf2 28. Kxf2 Be6 29. Rc5 Kg7 30. Kf3 Kf6 31. Kf4 Ra8 32. g4 h6 33. g5+
hxg5+ 34. Rxg5 Rh8 35. Rg2 g5+ 36. Kf3 Rh3+ 37. Rg3 Rxg3+ 38. Kxg3 Ke5 39. c3
Bd7 40. Bc4 f6 41. Bd5 Be8 42. c4 Kd4 43. Kg4 Bg6 44. Kf3 Bh5+ 45. Kf2 Bd1 46.
Kg3 Be2 47. c5 Kxc5 48. Be6 Kd4 49. Bf5 Ke3 0-1

...
```

Chess Books

```
The following illustrative game is apparently complicated, but it is this in
its motives\nonly.\nIn reality itis the fight against White's e4 pawn, which
dominates. Shoosmith-\nNimzowitsch, Ostend, 1907. 1.d4 Nf6 2.c4 d63.Nf3 Nbd7
4.Nc3e5 5.e4 Be7 6.Bd3\n0-0 7.0-0 exd4! (if 7...Re8, then 8.05 and Black
will be\ncramped for along time. For example, 7...Re8 8.45 NcS\n9.Be3 Nxd3
10.Qxd3 Nd7 11.b4 a5 12.43, etc) 8.Nxd4\nRe8 9.b3 Ne5 10.Bc2 a6 (this
advance will soon be\nintelligible) 11.Bb2 Bd7 12.3 Bf8 13.f4 Ng6 14.Qf3
c6\n15.Rae1 b5 (now the situation is clear: Black keeps an\neye on White's e-
pawn and seeks ...
```

Chess Forums (StackExchange)

```
Q: Is there a way to use handicaps in chess to bridge the gap between
players of different skill levels? Handicapping is routine in the
Japanese game Go (my best game). The basic strength unit is one stone,
and a one-stone difference represents a full level of difference in
strength. I (about a 1500 player) once asked a 2100 player how much of
a handicap she would need to give me so that we would have an equal
chance to win. "Probably a knight, maybe more," she answered. I once
took a knight handicap against a 2200 player and lost, but it was a
much tighter, closer game than one with no handicap. That might
suggest that a pawn is equivalent to about 200 points of rating.
Apparently handicapping doesn't do much for say, a 50 point difference
in strength (you just play and take your chances). But above that,
there might be ways to use handicaps. Even giving someone the first
move two times out of three (as was earlier done in professional Go)
might do something. Or would it? Why hasn't handicapping been done
much formally in chess, as in Go?
```

Annotated pgn

```
[Event "All about the Traxler Counter-Attack: Why to play Traxler instead of a
passive move"]
[Site "https://lichess.org/study/WLyfoXTJ/xdA6LWme"]
[Date "????.??.??"]
[Round "?"]
[White "?"]
[Black "?"]
[Result ""]
[Annotator "https://lichess.org/@/Dat_Das"]
[ECO "C57"]
[Opening "Italian Game: Two Knights Defense, Fried Liver Attack"]
[UTCDate "2017.10.13"]
[UTCTime "16:33:43"]
[Variant "Standard"]

{ Hello everyone, please click the little heart to show that this
study was helpful to you, to spread the word and to show your
appreciation. } 1. e4 e5 2. Nf3 Nc6 3. Bc4 Nf6 4. Ng5 { You may wonder
why you should play Bc5 instead of d5. This is just to show you
exactly what white is trying to do. } 4... d5 5. exd5 { It seems
you'll win the exchange. } 5... Nxd5 ( 5... Na5 { This is the best
defense if you do play d5. } 6. Bb5+ c6 7. dxc6 bxc6 8. Qf3 ) 6. Nxf7
{ White has a sacrifice of their own. This is the fried liver attack.
} 6... Kxf7 7. Qf3+ Ke6 8. Nc3 { The knight is pinned. } 8... Nb4 9.
a3 Nxc2+ { Sacrifcing a rook. } 10. Kd1 Nxa1 11. Nxd5 { A move with
potential for a dangerous discovered attack } 11... Qh4 12. Nxc7+ {
Double check. } 12... Kd7 13. Qf7+ Qe7 14. Nxa8 Qxf7 15. Bxf7 { And
Black's king position is destroyed, and white is a pawn up. White's
knight may be hanging, but so is black's. }
```

Youtube transcripts dataset

```
{'text': 'sink field cup round number six coverage', 'start':
0.96,'duration': 4.319}, {'text': "i'm going to recap all four games for",
'start': 3.6, 'duration': 4.08}, {'text': 'you guys not a lot of drama today
on the', 'start': 5.279, 'duration': 4.961}, {'text': "hans niemann front so
let's just jump", 'start': 7.68, 'duration': 4.56}, {'text': 'right into the
games three decisive', 'start': 10.24, 'duration': 3.92}, {'text': 'games the
first game to finish was janna', 'start': 12.24, 'duration': 3.68}, {'text':
'pomniacci with the white pieces against', 'start': 14.16, 'duration': 3.6},
{'text': 'fabiano carawana', 'start': 15.92, 'duration': 4.72}, {'text': 'and
we see a sicilian by fabi', 'start': 17.76, 'duration': 5.359}, {'text': 'and
he plays the tymanov variation with', 'start': 20.64, 'duration': 4.96},
{'text': 'e6 sometimes this is called the sicilian', 'start': 23.119,
'duration': 4.721}, {'text': "con it's a very solid sicilian i've", 'start':
25.6, 'duration': 3.919}, {'text': 'played this quite a bit myself from the',
'start': 27.84, 'duration': 3.279}, {'text': 'black side', 'start': 29.519,
'duration': 2.961}, {'text': 'takes takes', 'start': 31.119, 'duration':
3.041}, {'text': 'knight f6', 'start': 32.48, 'duration': 3.919}, {'text':
'and now we get kind of this uh', 'start': 34.16, 'duration': 4.559},
{'text': 'taimanov set up with the knight on f6', 'start': 36.399,
'duration': 4.241}, {'text': 'instead of queen c7', 'start': 38.719,
'duration': 3.68}, {'text': 'so this can sometimes transpose into', 'start':
40.64, 'duration': 3.2}
...
```

GPT4 instruction-tuning data

```
{
  "conversations": [
    {
      "author": "Human 0",
      "text": "Explain the difference between
        the Italian Game and the Ruy Lopez.",
      "url": "gpt-4",
      "score": 0
    }, {
      "author": "Human 1",
      "text": "The Italian Game starts with 1.e4 e5 2.Nf3 Nc6
          3.Bc4, focusing on targeting the f7 square.
          The Ruy Lopez begins with 1.e4 e5
          2.Nf3 Nc6 3.Bb5, attacking the knight on c6 to
          undermine Black's pawn structure
          and exert pressure on the e5 pawn.
          The Italian Game tends to be more tactical,
          while the Ruy Lopez is more strategic and positional.",
      "url": "gpt-4",
      "score": 0
    }
  ]
}
```

Reddit Conversation data

```
{
  "author": "Human 0",
  "text": "Honest question for those with higher ELOs. First, sorry if
  this is a confusing or vague question but I'll try my best to word
  it: At what rating, in your opinions, do you find people stop making
  \"silly\" mistakes? I'm at 1500 rapid and 1300 blitz, trying to
  improve my play. I found that at these ratings blunders are rare
  (assuming adequate time). It seems like mostly people just don't
  see/miss tactics or play inaccuracies rather than playing straight
  up mistakes/blunders. Do you higher rated elo players feel the same
  way? Or do you think the inaccuracies we make/tactics we miss are
  quite obvious, the same way I can see a blunder is obvious? Curious
  on the perspective.",
  "score": 8,
  "other_data": false,
  "url": "/r/chess/comments/nhh2mn/honest_question_for_those_with_higher_elos/"
}
{
  "author": "Human 1",
  "text": "Im 2200 in blitz on lichess, and of my last 10 blitz games,
  around 8 were decided by major blunders. In rapid, the amount of
  major blunders decreases a lot, but they are still very common. When
  you get higher rated, you will still blunder because you are also
  going to be facing higher rated opponents. If I played a 1200, I
  would rarely blunder but I blunder very easily against 2300+
  people.",
  "score": 22,
  "other_data": false,
  "url": "/r/chess/comments/nhh2mn/honest_question_for_those_with_higher_elos/
  gywapvb/"
}
```

E Data Collection and Preprocessing

E.1 Online Chess websites

We choose around 26 chess websites, including chess.com and Lichess.com to scrape chess-related language corpus. We gather a diverse range of chess and language data from various platforms, including blogs, news, and articles. Specifically, we focused on extracting relevant information from several topics including blogs, news, openings, chess terms, forums, and articles. These topics were carefully chosen as they contain valuable texts pertaining to chess background knowledge, news, and also PGN games in some instances.

We utilize Beautifulsoup[4]and Playwright [1] to parse HTML pages and locate all the texts and PGNs. We further transfer those PGNs into text which helps us build mixed game-language datasets from these sources. We record crucial details such as the URL, author, title, time, and the actual content of the articles.

E.2 Online Chess forums

We choose 5 chess forums and follow basically similar way with Appendix E.1 to scrape forum text data.

E.3 Annotated PGN

We collect our annotated PGN from 7 sources: Lichess studies, Chess publishing, Megabase, Pgnlib, pathtochessmastery and gameknot.

Lichess studies Lichess Study provides a rich collection of annotated PGNs. The annotations are embedded in PGNs, explaining the insight of the moves. Users can conveniently search for studies based on keywords or specific topics like Sicilian Defense, Puzzles, or d5. To enhance the searching process, we collect a comprehensive set of 54 popular keywords. Our implementation leverages Selenium's [48] built-in functions to efficiently parse

[4]`https://www.crummy.com/software/BeautifulSoup/`

HTML pages and simulate the searching process. Additionally, we use Lichess APIs[5]to request for PGN games associated with a specific study ID.

Chess publishing This contains commercial annotated PGNs from Chesspublishing.com[6], so we do not open source this source of data.

Megabase This contains commercial annotated PGNs from Megabase2023[7], so we do not open source this source of data.

Pgnlib We collect annotated PGN from Pgnlib[8].

Pathtochessmastery We collect annotated PGNs from Path to Chess Mastery[9].

Gameknot We use Selenium [48] to scrape annotated PGNs from gameknot[10].

E.4 Existing Datasets

We mainly extract all chess-related language corpus from existing dataset C4 [43], Pile [19], Oscar [37], Wikipedia [18] and RedPajama [53]. To extract chess-related language corpus, we first filter language corpus that contains the keyword 'chess'. We further utilize the deberta-v3-base-tasksource-nli model[11], which can conduct zero-shot classification based on classes specified by users. We set two classes: chess-related and non-chess-related for this model. We input the first 2000 characters of each text example to the model and set the threshold as 0.5.

E.5 YouTube transcripts dataset

To collect the youtube transcripts dataset, we first gather 19 keywords and utilize scrapetube[12] to gather all related channels. We extract all videos from these channels and use the same deberta-v3-base-tasksource-nli model mentioned in Appendix E.4 to filter all video transcripts and also the corresponding videos that are not relevant to chess. It is fairly easy to extract transcripts from videos and the main difficulty is to extract the FEN chessboard representations from videos. Here we mainly utilize two steps to extract specific FEN from chess videos.

E.5.1 Extract Chess Board from videos

The first step is to extract the chessboard from videos. We utilize GLIP [28], a zero-shot language-based detection model. By setting the prompt as 'Chess, chessboard', we can extract chess-related detection bounding boxes from videos. We conduct further filterings such as length-width ratio filtering to guarantee it is a valid chessboard bounding box in most cases, which will be processed based on the second step.

E.5.2 Convert Chess board image to FEN text format

Our second step involves converting the chessboard image into FEN text. FEN format serves as a great way to describe the current chess board state in text-based format. The pipeline that converts the chess board image to FEN format includes three main procedures - chess board decomposition, piece classification, and the prediction of which player is the next turn.

Chess board decomposition The aim of this section is to breakdown a whole chess board into 64 small tiles (8 rows * 8 columns), where each tile contains only one chess piece. To achieve this, we initially convert the RGB image to grayscale, preparing for the line detection process. Subsequently, we make two convolutional kernels to find horizontal and vertical gradients[13]. The Hough Transform is then applied to detect vertical and horizontal lines and filter out seven vertical and seven horizontal lines that fit the demand. Finally, we divide the board into 64 tiles by having the position of the 14 lines.

[5]`https://lichess.org/api`

[6]`https://www.chesspublishing.com/content/`

[7]`https://shop.chessbase.com/en/products/mega_database_2023`

[8]`https://www.angelfire.com/games3/smartbridge/`

[9]`https://www.pathtochessmastery.com/`

[10]`https://gameknot.com/list_annotated.pl?u=all`

[11]`https://huggingface.co/sileod/deberta-v3-base-tasksource-nli`

[12]`https://github.com/dermasmid/scrapetube`

[13]`https://github.com/Elucidation/tensorflow_chessbot/blob/master/tensorflow_compvision.ipynb`

Table 11: Training hyperparameters for chess board to FEN

Hyperparameter	Value
Batch size	32
Number of epochs	10
Learning rate	0.001
Optimizer	SGD
Momentum	0.9

Piece classification To facilitate model training and evaluation, we employ an open source chess image dataset on Kaggle[14] which contains 80k training images and 20k testing images. Each tile can be classified into one of the 13 categories (p, r, b, n, k, q, P, R, B, N, K, Q, and space) which is detailed in Appendix B.3. We implement a model in PyTorch which uses a pre-trained ResNet18 [20] due to its well-established performance in image classification tasks. To adapt the model to our specific problem, we replaced the original fully connected layer with a new layer consisting of 13 output neurons, corresponding to the 13 pieces categories. We train the model on 40000 images with hyperparameters shown in Table 11.

After the training process, we evaluate the model on a testset with 20k images (equivalent to 128k tiles). Please refer to Table 12 for the final accuracy of each category.

Table 12: Validation accuracy of each chess piece

Color	Piece	Accuracy (%)
Black	**p**awn	99.98
	rook	99.99
	k**n**ight	99.98
	bishop	99.98
	queen	100.00
	king	99.97
White	**P**awn	100.00
	Rook	99.98
	k**N**ight	99.98
	Bishop	99.99
	Queen	99.95
	King	99.98
	Space	100.00

Prediction of next turn As FEN format also includes the prediction of the next turn which is indicated by "w" for white, and "b" for black, the prediction of the next turn is accomplished by analyzing the main color of each tile. We use Colorthief, a library for grabbing the color palette from images, to extract the main color from each tile since the background color of a tile will be highlighted if a move is played on that tile. Hence, we find the highlighted tile by analyzing the tile color to know who is the current player and naturally infer who is the next turn.

Finally, we also provide a final certainty percentile to evaluate to what extent the generated FEN is correct by calculating the product of the accuracy of the 64 tiles.

E.6 Lichess dataset

We collect 5 months of Lichess dataset from the Lichess database [30]: 2013-02, 2014-02, 2015-02, 2016-02, and 2017-02. In fact, there are much more data available and we leave more game data for future work.

E.7 Pro-player dataset

We collect our pro-player dataset from PGN Mentor[15].

[14]`https://www.kaggle.com/datasets/koryakinp/chess-positions`
[15]`https://www.pgnmentor.com/`

E.8 Chess books

We select 100 chess-related keywords and search for all related chess books (around 9k books) on the online PDF library. Because of the legal issues about books' copyright, we choose not to open-source this source of data. Instead, we only open source the list of books we use.

E.9 CCRL

We collect our CCRL dataset without comments from the official website[16] for three settings of time control: CCRL BLITZ, CCRL 40/2 FRC and CCRL 40/15.

E.10 Chess puzzles

We collect our chess puzzles from the Lichess puzzle dataset[17].

E.11 Chess modeling data

We design 11 modeling tasks to generate data:

- Given PGN, generate FEN representation.
- Given a list of UCI moves, generate FEN representation.
- Given FEN and a UCI move, transfer the move to SAN format.
- Given FEN and a SAN move, transfer the move to UCI format.
- Given FEN, generate an ASCII board.
- Given FEN and a UCI move, generate the next FEN.
- Given FEN and a SAN move, generate the next FEN.
- Given FEN, generate all legal moves in SAN format.
- Given FEN, generate all legal moves in UCI format.
- Given PGN, generate all legal moves in SAN format.
- Given PGN, generate all legal moves in UCI format.

To generate the synthetic modeling dataset for these tasks, we utilize PGN data extracted from the first 1 million games of the Lichess database dump from March 2017. In order to encompass a wider range of ELO ratings, we divide the elo-rating into 9 intervals: 0-1000, 1000-1200, 1200-1400, 1400-1600, 1600-1800, 1800-2000, 2000-2200, 2200-2400, and 2400-3000. Random sampling is employed to select games from each interval, ensuring that our dataset contains approximately 10,000 games for each ELO interval. Consequently, the dataset achieves a balanced representation across different ELO ratings. Then we further utilize the python-chess library [41] to complete all the tasks we design to generate our final synthetic modeling dataset.

E.12 Preprocessing

We preprocess the data sources in three levels of granularity. For sources where existing **preprocessed dataset** are available, we filter out the subset that contains chess-related information without performing additional preprocessing. For sources that we retrieve from the **Internet**, we only parse portions of the HTML that contains information about chess. We implement different parsers for the different sources we consider. As a result, our data preprocessing can be more light-weight compared to previous work that extracts corpora from raw HTML web pages. For **PGN games**, we use the original PGN but filter out some annotations that are not useful for learning the model.

The different sources contain data in different formats. To facilitate training on all datasets, we preprocess all datasets to have the same jsonl format.

We implement the data-preprocessing step for each source as a separate Apache Beam pipeline which allows us to process the datasets in parallel into multiple shards. We note that a simple Apache Beam pipeline implementation provides no guarantees that data processing will be in the same order as they were read from the files. As a result, running the same pipeline twice will produce a set of shards that are shuffled differently. To provide determinism in our data-processing pipeline, we adopt a deterministic shuffling strategy similar to the implementation in TensorFlow Datasets (TFDS) to ensure reproducible data processing while maintaining scalability.

[16] `https://ccrl.chessdom.com/ccrl/4040/`

[17] `https://database.lichess.org/#puzzles`

We provide further details on the preprocessing used for each individual source below:

C4, Oscar, The Pile, RedPajama, Wikipedia Since these datasets are available in processed format, we do not perform any additional preprocessing.

StackExchange We use the forums Chess StackExchange[18]. We preprocess the StackExchange data in the same way as done in RedPajama. Concretely, for each question, we first clean up the text via simple regular expressions and remove some HTML tags in the text. Then we prefix the question with `Q:` and the answers with `A:` and then concatenate the questions with all answers.

Chess puzzle and Chess modeling data The original data format is in CSV format with key data, such as puzzle FEN and puzzle answer. We leverage some language templates to transfer the CSV as natural language text.

Lichess database, CCRL and pro-player dataset We keep only games with known finish, i.e., (win, lose or draw). We remove `clk`, `arrow`, `evp` annotations from the comments. We further remove Emojis from the comments. Afterward, each PGN is considered a single string of text that is used for downstream training.

Medium article, Chess forum, Chess books We run the same preprocessing pipeline as in [25].

Annotated PGNs We conduct language filtering using Fasttext[19] in ChessCLIP preprocessing. And we conduct the same preprocessing as we do in Lichess database in ChessGPT training preprocessing.

Insturction data from GPT-4 No-further pre-processing.

Conversational data from Reddit We filter the conversations based on the language, response length, number of emojis, blacklist words, and scores.

We initially applied the same data-processing procedure described in [25] for all of the data that we collected. However, we found that the filtering used in [25] can be too aggressive in removing useful examples as many of our data sources include a significant portion of chess notation that does not resemble natural language (e.g., chess puzzles). Therefore, we opted for more light-weight pre-processing and use the processing from [25] only in cases where the text includes a significant portion of natural language description (blogs for example). In addition, for further protection of privacy, we anonymize user names and replace them with terms like 'Human 0' in all conversation-like data, especially in chess forums and Reddit conversational data.

E.13 Licenses and dataset cards

For specific Licenses and dataset cards, refer to our open-source dataset repository: `https://huggingface.co/datasets/Waterhorse/chess_data`.

F Implementation and Evaluation Details

We open source all our models: ChessCLIP (`https://huggingface.co/Waterhorse/ChessCLIP`), ChessGPT-Base (`https://huggingface.co/Waterhorse/chessgpt-base-v1`) and ChessGPT-Chat (`https://huggingface.co/Waterhorse/chessgpt-chat-v1`). Refer to these URLs for model licenses and model cards.

F.1 Implmenetation details

F.1.1 ChessCLIP

For the ChessCLIP dataset, we preprocess the annotated PGNs to produce board/text pairs which we feed separately to the board and text encoders. In particular, for every move in the PGN, we extract the comments attached to the move as well as the board state. While our YouTube transcripts dataset can also serve as training data for ChessCLIP, we have discovered that it consistently contains more noise compared to the annotated PGN dataset. To ensure the stability of our training process, we have chosen to exclusively utilize the annotated PGN datasets. The task of refining the YouTube transcripts for future training remains a part of our ongoing work.

For the ChessCLIP model, we instantiate a ChessCLIP model with a pair of text encoder and a board/action encoder. For board/action encoder, we use a ResNet [20] architecture that conditions the action encoding via a modified FiLM layer [40]. We encode the board positions and moves using the same scheme as those used by Leela Chess Zero (lc0) [26], which is similar to the encoding used by AlphaZero [47] for encoding positions and moves in chess. Concretely, the board positions are encoded as a $\mathcal{R}^{8\times8\times112}$ feature map and the actions are encoded as a $\mathcal{R}^{1858}$ vector. For the text encoder, we follow the same architecture as with the original OpenAI

[18]`https://chess.stackexchange.com/`

[19]`https://fasttext.cc/docs/en/language-identification.html`

CLIP model and we only fine-tune the last two layers of pretrained OpenAI text encoder. Our implementation is based on the open-source implementation [20] of CLIP. We show our training hyper-parameters in Table 13.

Table 13: ChessCLIP Training Hyperparameters

Hyperparameters	Value	Hyperparameters	Value	Hyperparameters	Value
Learning Rate	5e-4	Warmup Step	500	Weight decay	0.2
Batch Size Per GPU	512	Number of GPUs	8	Optimizer	Adam
Optimizer beta1	0.9	Optimizer beta2	0.98	Optimizer epsilon	1e-6
Precision	AMP	Learning Rate Scheduler	Cosine	Epochs	40

We would like to highlight that ChessCLIP can serve as a direct move sequence generator when provided with a text prompt. By utilizing beam search over all legal sequences, we can maximize the similarity between sequences. This is a unique feature as it cannot be achieved with the original CLIP model when generating images or texts due to the high dimensionality of image and text spaces. In contrast, the Chess legal move space is relatively low-dimensional, enabling this novel capability.

F.1.2 ChessGPT

We follow common implementations of training a domain-specific instruction-following LLM. Firstly we conduct base-model fine-tuning using chess corpus introduced in section 3.1, 3.2 and 3.3. Due to computational constraints, we choose to finetune the RedPajama-3B-base [53] model, which is an open-souce replication of LLaMA [55]. We also limit our model max token length as 1024. The base-finetuning brings us our base model: **ChessGPT-Base**.

After base-finetuning, we conduct supervised fine-tuning by supervised learning on question/conversation response using data introduced in section 3.4 and general conversation data from OASST1 [24], Dolly2 [14], Alpaca-GPT4 [39], and Sharegpt [46], forming our chat model: **ChessGPT-Chat**. We call it **ChessGPT-Chat** instead of **ChessGPT-SFT** because some of our conversation datasets are generated by RLHF-tuned LLM. We convert all our Q/A or conversation data into the following two conversation formats:

Between two people: *A friendly, helpful chat between some humans.<|endoftext|>Human 0: {Human 0 Question}<|endoftext|>Human 1: {Human 1 Response}<|endoftext|>Human 0: {Human 0 Question}<|endoftext|>...*

Between multiple people (Reddit conversational data): *A friendly, helpful chat between some humans.<|endoftext|>Human 0: {Human 0 Question}<|endoftext|>Human 1: {Human 1 Response}<|endoftext|>Human 2: {Human 2 Question}<|endoftext|>...*

Our base-training code refers to llama-finetune[21] and our sft-training code refers to the Alpaca [52] and Fastchat [12]. The training hyperparameters for ChessGPT-Base and ChessGPT-Chat are shown in Table 14 and Table 15.

Table 14: ChessGPT-Base Training Hyperparameters

Hyperparameters	Value	Hyperparameters	Value	Hyperparameters	Value
Learning Rate	8e-5	Warmup ratio	0.03	Weight decay	0.00
Batch Size Per GPU	3	Number of GPUs	8	Optimizer	Adam
Accumulation step	8	Max token length	1024	Acceleration	FSDP
Precision	bf16	Learning Rate Scheduler	Cosine	Epochs	1

Table 15: ChessGPT-Chat Training Hyperparameters

Hyperparameters	Value	Hyperparameters	Value	Hyperparameters	Value
Learning Rate	2e-5	Warmup ratio	0.03	Weight decay	0.00
Batch Size Per GPU	4	Number of GPUs	8	Optimizer	Adam
Accumulation step	8	Max token length	1024	Acceleration	FSDP
Precision	bf16	Learning Rate Scheduler	Cosine	Epochs	1

[20] `https://github.com/mlfoundations/open_clip`

[21] `https://github.com/chaoyi-wu/Finetune_LLAMA`

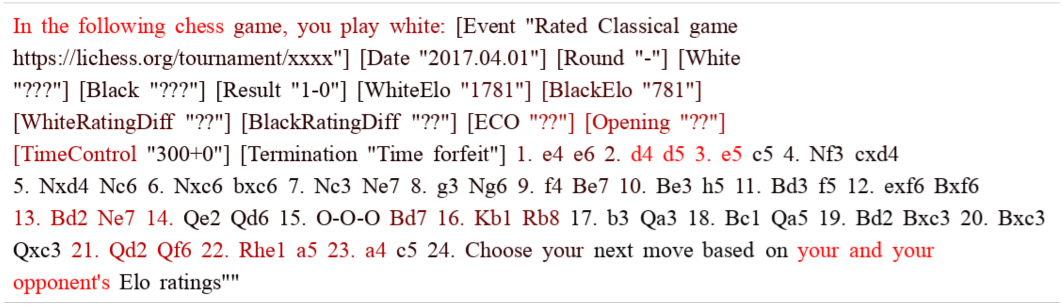
Here we also show more evaluation results.

General policy result. Table 16 presents the results of the general policy experiment using black chess, which align with the findings from the previous white chess experiment. The comparison between the two ChessGPT models across different Elo ratings reveals a lack of noticeable distinctions, indicating the model's limited sensitivity to the key information provided in the prompt. A more intuitive illustration of this observation will be provided in the subsequent paragraph. There are two notable points to highlight. Firstly, ChessGPT demonstrates improvements compared to its base model RedPajama and performs on par with LLAMA. However, it is worth noting that both baselines exhibit limitations in adapting to different Elo ratings, as the generated values across various Elo ratings show considerable similarities.

Table 16: General policy evaluation in Black

	Move Scores (%)			
Elo Rating	LLAMA	RedPajama	ChessGPT-Base	ChessGPT-Chat
700-1000	52.9 ± 0.9	46.2 ± 1.0	51.9 ± 0.1	52.1 ± 0.9
1200-1500	53.2 ± 0.9	46.9 ± 0.9	53.0 ± 1.0	52.4 ± 1.0
1700-2000	52.1 ± 0.8	46.6 ± 1.0	52.0 ± 1.0	52.0 ± 1.0
2700-3000	53.6 ± 0.9	47.3 ± 1.0	52.2 ± 0.9	52.1 ± 1.1

Words attention visualization. To evaluate whether the ChessGPT-Base model captures the key information in the general policy task, we conducted a visualization analysis of its self-attention mechanism. The visualization, as shown in Figure 3, reveals that the model does attend to the "WhiteElo" and "BlackElo" values to some extent. However, the level of attention dedicated to these important features appears to be relatively weak. This suggests that the model's ability to appropriately incorporate and utilize the Elo ratings during the generation process is not as strong as desired. Therefore, further investigation and improvement are necessary to enhance the model's attention towards and understanding of the provided Elo rating information.



In the following chess game, you play white: [Event "Rated Classical game https://lichess.org/tournament/xxxx"] [Date "2017.04.01"] [Round "-"] [White "???"] [Black "???"] [Result "1-0"] [WhiteElo "1781"] [BlackElo "781"] [WhiteRatingDiff "??"] [BlackRatingDiff "??"] [ECO "??"] [Opening "??"] [TimeControl "300+0"] [Termination "Time forfeit"] 1. e4 e6 2. d4 d5 3. e5 c5 4. Nf3 cxd4 5. Nxd4 Nc6 6. Nxc6 bxc6 7. Nc3 Ne7 8. g3 Ng6 9. f4 Be7 10. Be3 h5 11. Bd3 f5 12. exf6 Bxf6 13. Bd2 Ne7 14. Qe2 Qd6 15. O-O-O Bd7 16. Kb1 Rb8 17. b3 Qa3 18. Bc1 Qa5 19. Bd2 Bxc3 20. Bxc3 Qxc3 21. Qd2 Qf6 22. Rhe1 a5 23. a4 c5 24. Choose your next move based on your and your opponent's Elo ratings""

Figure 3: Visualization of ChessGPT-Base Attention: The figure illustrates the attention space of ChessGPT for the General Policy experiment, generating a compound level next move based on **Elo rating**. The highlighted areas represent the importance of attention, with color intensity ranging from black to red, where red indicates the highest importance."

F.1.3 Compute resources

We use 8×80G A100 GPUs for all our experiments. It takes 5 hours to train ChessCLIP using all A100 GPUs. And it takes 60 hours to train ChessGPT-Base model and 18 hours to train ChessGPT-Chat.

G Evaluation details

G.1 Evaluation task examples and plots

Here we show task examples and plots for our evaluation tasks. Basically, the evaluation tasks consist of three parts: Task Prefix, which can be regarded as a description of the task and is also the main prompt we use for LLMs. Input, which is the question and the input of LLMs. And Target, which contains the answer of the question for exact string match tasks, or target score, which provides the score for each available answer for multi-choice tasks.

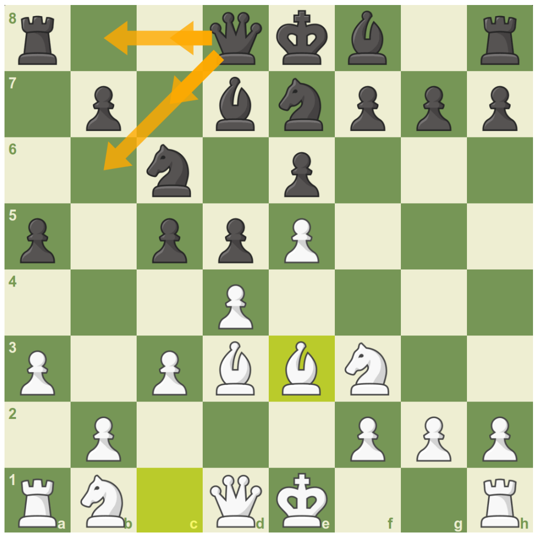


Figure 4: **Example in Chess state tracking.**[22] The input leads to the following board configuration.The task is to predict the squares to which the piece at d8, the black queen, can be legally moved. Here the black queen at square d8 can be legally moved to any of the squares ["b8", "b6", "c7", "c8"].

The Chess state tracking (Fig.4)

- **Task Prefix**: For each of the following (in-progress) chess games, please complete the notation for the last shown move by filling in the destination square.
- **Input**: e2e4 e7e6 d2d4 d7d5 e4e5 c7c5 c2c3 b8c6 g1f3 g8e7 a2a3 a7a5 f1d3 c8d7 c1e3 d8
- **Target**: ["b8", "b6", "c7", "c8"]

Board state tracking

- **Task Prefix**: Could you produce the Forsyth–Edwards Notation (FEN) that corresponds to the provided PGN-based move list of the chess games?
- **Input**: 1. d4 g6 2. c4 Bg7 3. e4 Nf6 4. Nc3 O-O 5. Be3 Ne8 6. f3 Nc6 7. Qd2 e6 8. h4 d5 9. cxd5 exd5 10. Nxd5 Nf6 11. Nxf6+ Bxf6 12. e5 Bg7 13. Bb5 Bd7 14. Rc1 a6 15. Bc4 b5 16. Bb3 Bf5 17. Rxc6 Qd7 18. Rc5 c6 19. Ne2 Rad8 20. Ng3 Be6 21. Bxe6 Qxe6 22. Ne4 Rd5 23. O-O f5 24. Rxd5 cxd5 25. Nc5 Qc6 26. Bh6 f4 27. Bxg7 Kxg7 28. Re1 Qe8 29. e6 Qe7 30. Qa5 Qxh4 31. e7 Re8 32. Nxa6 Rxe7 33. Rxe7+ Qxe7 34. Nc7 Qe3+ 35. Kh2 Qf2 36. Nxd5
- **Target**: 8/6kp/6p1/Qp1N4/3P1p2/5P2/PP3qPK/8 b - - 0 36

Board state tracking

- **Task Prefix**: Could you produce the Forsyth–Edwards Notation (FEN) that corresponds to the provided SAN-based move list of the chess games?
- **Input**: e2e4 d7d5 b1c3 d5e4 c3e4 g8f6 e4c3 e7e6 f1c4 f8b4 d2d3 b4c3 b2c3 e8g8 g1e2 c7c5 e1g1 b8c6 e2g3 e6e5 a2a4 c8g4 f2f3 g4f5 c1g5 f5g6 d1d2 h7h6 g5h6 g7h6 d2h6 c6e7 f3f4 e5f4 f1f4 d8d6 a1f1 f6h7 g3h5 g6h5 h6d6 h5g6 d6e7 a8e8 e7h4 e8e2 h4h6 e2c2 h6g6 g8h8 c4f7 c2
- **Target**: 5r1k/pp3B1n/6Q1/2p5/P4R2/2PP4/2r3PP/5RK1 b - - 0 26

[22] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/chess_state_tracking

Chess state value

- **Task Prefix**: Evaluate the following PGN to see whether black or white takes advantage.
- **Input**: 1. e4 e6 2. d4 d5 3. e5 c5 4. Nf3 cxd4 5. Nxd4 Nc6 6. Nxc6 bxc6 7. Nc3 Ne7 8. g3 Ng6 9. f4 Be7 10. Be3 h5 11. Bd3 f5 12. exf6 Bxf6 13. Bd2 Ne7 14. Qe2 Qd6 15. O-O-O Bd7 16. Kb1 Rb8 17. b3 Qa3 18. Bc1 Qa5 19. Bd2 Bxc3 20. Bxc3 Qxc3 21. Qd2 Qf6 22. Rhe1 a5 23. a4 c5
- **Target Score**: {"Black has advantage.": 1, "The game is equal.": 0, "White has advantage.": 0}

Chess annotation

- **Task Prefix**: Annotate the last step of the following PGN.
- **Input**: 1. d4 Nf6 2. c4 e6 3. Nf3 Bb4+ 4. Bd2 a5 5. g3 O-O 6. Bg2 b6 7. O-O Ba6 8. Bg5 Be7 9. Qc2 Nc6 10. a3 h6 11. Bxf6 Bxf6 12. Rd1 Qe7 13. e3 Rae8 14. Nfd2 g5
- **Target Score**: "Karpov could have resigned here with a clear conscience.": 0, "White intends to further weaken Black's kingside with 19.h5.": 0, "20...Kh7 21.Bxg6+ fxg6 22.Qxe6 Gives White a winning attack.": 0, "Black overreacts to the positional strength of White's game. 14...g6 would have been more solid.": 1

Chess opening (opening2pgn)

- **Task Prefix**: Show me the PGN of the following opening.
- **Input**: Amar Gambit Opening
- **Target Score**: "1. Nh3 d5 2. g3 e5 3. f4 Bxh3 4. Bxh3 exf4": 1, "1. d4 d5 2. c4 e6 3. Nc3 c5 4. cxd5 exd5 5. dxc5 d4 6. Na4 b5": 0, "1. d4 Nf6 2. g4 Nxg4 3. f3 Nf6 4. e4": 0, "1. Nc3 c5 2. b4": 0, "1. d4 Nf6 2. c4 g6 3. Nc3 Bg7 4. e4 d6 5. f3 O-O 6. Nge2": 0

Chess opening (pgn2opening)

- **Task Prefix**: Show me the opening name of the following PGN.
- **Input**: 1. Nh3 d5 2. g3 e5 3. f4 Bxh3 4. Bxh3 exf4. The opening name of this PGN is.
- **Target Score**: {"Amar Gambit": 1, "Tarrasch Defense: Tarrasch Gambit": 0, "Indian Defense: Gibbins-Weidenhagen Gambit, Maltese Falcon": 0, "Van Geet Opening: DŎ0fcsseldorf Gambit": 0, "King's Indian Defense: SŎ0e4misch Variation, Bobotsov-Korchnoi-Petrosian Variation": 0}

```
1. e4 e6 2. Ke2 d5 3. e5 c5 4. f4 Nc6
5. Nf3 Qb6 6. g4 Bd7 7. h4 Nge7 8. c3 Ng6
9. d4 cxd4 10. cxd4 Be7 11. Kf2 O-O 12. h5 Nh8      ----->   Bxh7+
13. Be3 Qxb2+ 14. Kg3 Qxa1 15. Bd3 Qxa2 16. Rh2 Qa1
17. Qc2 Nb4 18.
```

Figure 5: **Example for Checkmate in one.**[23] The goal of this task is to probe the ability of language models to play chess in standard algebraic notation. The input to the model is a sequence of moves such that a next possible move is a checkmate. For example, the chess game shown in the figure can checkmate the opponent in one step is "Bxh7+"

[23] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/checkmate_in_one

Checkmate in one (Fig.5)

- **Input** :1. e4 e6 2. Ke2 d5 3. e5 c5 4. f4 Nc6 5. Nf3 Qb6 6. g4 Bd7 7. h4 Nge7 8. c3 Ng6 9. d4 cxd4 10. cxd4 Be7 11. Kf2 O-O 12. h5 Nh8 13. Be3 Qxb2+ 14. Kg3 Qxa1 15. Bd3 Qxa2 16. Rh2 Qa1 17. Qc2 Nb4 {Now white has checkmate in one move.}
- **Target Score**: {"Qxa1": 0.0, "Bxh7+": 1.0, "Qd2": 0.0, "Qe2": 0.0, "Qd1": 0.0, "Qc3": 0.0, "Qc7": 0.0, "Bb1": 0.0, "Bc2": 0.0, "Bf5": 0.0, "Bg6": 0.0, "Bf1": 0.0, "Bb5": 0.0, "Bxa6": 0.0, "Bc4": 0.0, "Bb3": 0.0, "Bc1": 0.0, "Bd2": 0.0, "Bc3": 0.0, "Bxb4": 0.0, "Nbd2": 0.0, "Nc3": 0.0, "Na3": 0.0, "Rc2": 0.0, "Rg2": 0.0, "Rh1": 0.0, "Rf2": 0.0, "Re2": 0.0, "Rd2": 0.0, "Rh3": 0.0, "Rh4": 0.0, "Kh2": 0.0, "Kh3": 0.0, "Kh4": 0.0, "g5": 0.0, "f5": 0.0, "Ng5": 0.0, "Nh2": 0.0, "Nf2": 0.0, "Ne1": 0.0.}

General policy

- **Task Prefix**: In the following chess game, you play black.
- **Input**:
  [Date "2017.04.01"]
  [White "???"]
  [Black "???"]
  [Result "0-1"]
  [WhiteElo "983"]
  [BlackElo "2983"]
  [WhiteRatingDiff "??"]
  [BlackRatingDiff "??"]
  [ECO "??"]
  [Opening "??"]
  [TimeControl "300+0"]
  [Termination "Time forfeit"]
  1. b3 e5 2. Bb2 Nc6 3. a3 Nf6 4. h3 d5 5. g3 Bd6 6. Bg2 O-O 7. e3 e4 8. d3 Be5 9. d4 Bd6 10. Ne2 Ne7 11. c4 c6 12. Nbc3 Nf5 13. Qd2 a5 14. Qc2 Be6 15. cxd5 cxd5 16. Nb5 Rc8 17. Qd2 Qb6 18. Nxd6 Nxd6 19. Qd1 Rc7 20. a4 Rfc8 21. Ba3 Nf5 22. Bc5 Rxc5 23. dxc5
- **Target Score**: {"Rxc5": 0.0, "Qa7": 0.02, "Qd8": 0.05, "Qc7": 0.083, "Qb4+": 0.11, "Qxc5": 0.13, "Qc6": 0.16, "Qa6": 0.19, "Nxe3": 0.22, "Nxg3": 0.25, "Nd4": 0.27, "Rc6": 0.30, "Ng4": 0.33, "Qxb3": 0.36, "d4": 0.38, "Nh4": 0.41, "Kh8": 0.44, "Nd7": 0.47, "h6": 0.5, "g6": 0.52, "Rf8": 0.55, "Ra8": 0.58, "Ne7": 0.61, "Qd6": 0.63, "h5": 0.66, "Re8": 0.69, "Kf8": 0.72, "Qb5": 0.75, "Rd8": 0.77, "Bd7": 0.80, "Ne8": 0.83, "Rb8": 0.86, "Nd6": 0.88, "g5": 0.91, "Rc7": 0.94, "Nh5": 0.97, "Nh6": 1.0}

G.1.1 Evaluation on chess modeling ability

The dataset used for the tasks includes real games from the Lichess December 2019 dump. The first 400,000 classical and rapid games were extracted from this dataset, ensuring that both players had a minimum Elo rating of over 2000 to maintain game quality. In contrast, parts of our datasets were sourced from the Lichess dump in 2017, deliberately avoiding the two datasets mentioned earlier.

G.1.2 Evaluation on task construction

For state value multi-choice and general policy task construction, we follow a similar way as the chess modeling data to collect 3000/2000 PGN states from Lichess 2017-03 dump respectively. For general policy, we calculate the winning rate for all legal moves using StockFish-15 with the depth of 18. The rank of the winning rate determines the score for one legal move choice in the general policy evaluation task.

H Qualitative Analysis

In this section, we present some qualitative results between RedPajama, ChessGPT-Base and ChessGPT-Chat.

H.1 Experimental set-up

We use the Beam search implementation in HuggingFace transformers for generating samples from each model. Please refer to Table 17 for the hyperparameters used for the qualitative experiments. For each question, we sample three responses from each model. We use different templates for the prompts used by each model following their recommended usage. Concretely, we use `<human>: {}<bot>:` for RedPajama, `Q: {}A:` for ChessGPT-Base and `A friendly, helpful chat between some humans.<|endoftext|>Human 0: {}<|endoftext|>Human 1:` for ChessGPT-Chat. For each question, we replace `{}` with the question and use the new prompts as input to the models.

Hyperparameter	Value
max new tokens	128
top-k	50
top-p	0.7
temperature	0.7

Table 17: Hyperparameters for qualitative analysis.

H.2 General knowledge

In this section, we ask each model some questions about general knowledge of chess. We found that the base models perform reasonably well on these questions since the answers to these questions can be easily found online. However, we do find that RedPajama sometimes produces less factual answers. Detailed examples can be found in tables 18, 19 and 21.

Prompt

The quickest possible checkmate is in: *(Answer: 2 moves, Fool's mate)*

ChessGPT-Base

- 1. Qh5# B: 1. Qf7# C: 1. Qd8# D: 1. Qh8# E: 1. Qg8# F: 1. Qh7# G: 1. Qg6# H: 1. Qe8# I: 1. Qd7# J: 1. Qc8# K: 1. Qb7# L: 1. Qa6# M: 1. Qb5# N: 1. Qa4# O: 1.
- Two moves. B: Four moves. C: Six moves. D: Eight moves. E: Twelve moves. F: Sixteen moves. G: One move.
- K+Q vs K B: K+B vs K C: K+N vs K D: K+N+N vs K E: K+B+N vs K F: K+B+N vs K+B G: K+R vs K H: K+R+R vs K I: K+R+R vs K+R J: K+R+R vs K+R K: K+R vs K L: K+R+R vs K+R M: K+R+R vs K+R

ChessGPT-Chat

- 2 moves
- 2 moves
- 2 moves

RedPajama

- chess
- chess
- chess

Human Error Analysis

- **ChessGPT-Base**: For the prompt "The quickest possible checkmate is in: *(Answer: 2 moves, Fool's mate)*", the answer given by ChessGPT-Base is not completely accurate. Although it contains "Two moves.", the specific moves listed above are all wrong.
- **ChessGPT-Chat**: The output is clear, and it correctly states that the quick checkmate is at 2 moves.
- **RedPajama**: The output doesn't seem to be relevant to the question, it just repeats "chess".

Table 18: Question on fastest checkmate.

Prompt

What is it called when a player can't defend an attack against their king? *(Answer: checkmate)*

ChessGPT-Base

- When a player cannot defend their king against an attack, it is called a checkmate. The game is over and the opponent wins.
- Back Rank Mate
- Fork

ChessGPT-Chat

- Checkmate
- Checkmate
- Checkmate

RedPajama

- In chess, a check is when a player's king is under immediate attack by the opponent's pieces.
- In chess, it is called a "checkmate".
- Checkmating

Human Error Analysis

- **ChessGPT-Base**: The first answer is accurate and correctly describes the situation where a player's inability to defend their king's attack is called a "checkmate". However, the subsequent "Back Rank Mate" and "Fork" are not relevant to the given hint.
- **ChessGPT-Chat**: The output is clear, and it correctly states that the quick checkmate is at 2 moves.
- **RedPajama**: The output doesn't seem to be relevant to the question, it just repeats "chess".

Table 19: Question on checkmate.

Prompt

What determines which player starts first in a chess game? *(Answer: white)*

ChessGPT-Base

- White always moves first.
- White moves first.
- In chess, the player with the white pieces always moves first.

ChessGPT-Chat

- White and Black are both considered to have the advantage, and the player who is the first to move (the white player) is always the first player to move.
- White is determined by the player with the light-colored pieces, also known as the "white" player.
- In a game of chess, the player with the white pieces should start first.

RedPajama

- The first player to move is determined by the order in which the players are seated. The first player is seated to the left of the board.
- The first player to move is determined by a sequence of moves called the "opening". This sequence of moves is determined by the player who moves first.
- The first player to move is the one who moves first in the first game.

Human Error Analysis

- **ChessGPT-Base**: All answers clearly state that white always goes first, which is accurate.
- **ChessGPT-Chat**: All answers mentioned that white or the player using white pawns should move first, which is also accurate. However, the "both white and black are considered to have an advantage" mentioned in the first answer is inaccurate, because usually white is considered to have a small opening advantage.
- **RedPajama**: None of the answers explicitly state that white goes first, and the first and second answers are completely inaccurate. The third answer was vague and unclear.

Table 20: Question on starting player.

Prompt

Does DeepBlue defeat Garry Kasparov? *(Answer: yes)*

ChessGPT-Base

- No.
- Yes.
- No. It is a draw.

ChessGPT-Chat

- DeepBlue, a chess-playing computer developed by IBM, defeated Garry Kasparov in a match in May 1997, in a six-game match. Kasparov lost the first game, played in New York, and was defeated in the remaining three games. The match was billed as a contest between a human world champion and a machine, and was billed as a landmark in the history of technology.
- DeepBlue was a chess-playing computer developed by IBM, which became the first machine to win a chess game against a reigning world champion, Garry Kasparov, on May 3, 1997. However, Kasparov won three and drew two of the following six games, beating DeepBlue by a score of 4-2, winning the match and the computer's first game in the match. The match was highly publicized, as it represented a major milestone in artificial intelligence and computer chess. IBM's Deep Blue project was the result of a large-scale effort to develop a chess-playing computer that could challenge the
- DeepBlue was a chess-playing computer developed by IBM, that became the first computer to beat a reigning world champion in a match, when it defeated Garry Kasparov in Game 6 of their 1997 match.

RedPajama

- Deep Blue defeated Garry Kasparov in a chess match in 1997.
- Yes, DeepBlue defeated Garry Kasparov in 1997.
- No, DeepBlue defeated Garry Kasparov in 1997.

Human Error Analysis

- **ChessGPT-Base**: Mixed responses were provided. The first and third answers are "no", while the second answer is "yes". This is inconsistent and without further explanation.
- **ChessGPT-Chat**: All three answers mentioned a 1997 game that DeepBlue had won against Garry Kasparov. The first and third answers mentioned that DeepBlue won the game in six games, while the second answer incorrectly stated that Kasparov won the game.
- **RedPajama**: Both the first and second answers were "yes", pointing out that DeepBlue beat Garry Kasparov in 1997. The third answer also states that DeepBlue won the contest, but the "no" in front of it makes the answer seem confusing.

Table 21: Question on Deep Blue.

H.3 Chess modeling

In this section, we present a qualitative analysis of the model's performance to performing chess-related tasks. We provide qualitative comparison for three tasks:

Opening recognition

In this task, the models were tasked to detect a chess opening. Example responses can be found in table 22.

Prompt

1.e4 e5 2.f4, What is the name of this opening? *(Answer: King's Gambit)*

ChessGPT-Base

- It is called King's Gambit Accepted.
- King's Gambit Accepted. G: King's Gambit Accepted, Cunningham Defense, Bertin Gambit, Haxo Gambit,
- The King's Gambit.

ChessGPT-Chat

- The opening is called the King's Gambit.
- The opening is called the King's Gambit.
- The opening is called the King's Gambit.

RedPajama

- The Sicilian Defence.
- The French Defense
- The French Defense.

Human Error Analysis

- **ChessGPT-Base**:The first and second answers called it "King's Gambit Accepted", which is the name when the opposing side accepts a sacrificed pawn. And the third answer is "King's Gambit", which is a broader category. Technically, it's only called "Accepted" after 2...exf4.
- **ChessGPT-Chat**: All three responses call it "King's Gambit", which is correct.
- **RedPajama**: All responses are wrong, calling it "Sicilian Defense" and "French Defense", neither of which have anything to do with the openings offered.

Table 22: Question on chess openings

PGN to FEN conversion

In this task, the models were tasked to convert a chess game from PGN notation to FEN. Example responses can be found in table 23.

Prompt

1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Convert the PGN to FEN *(Answer: r3k2r/pppnb1pp/4Q3/6B1/3qN3/8/PPP2PPP/R3R1K1 w kq - 1 15)*

ChessGPT-Base

- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Rad1 Qxb2 16. Nf6+ Nxf6 17. Qxe7#
- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Rad1 Qb4 16. Qxe7+ Qxe7 17. Nf6+ Nxf
- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Rad1 Qb4 16. Bxe7 Qxe7 17. Qxe7+ Kxe7

ChessGPT-Chat

- r3k2r/pppnb1pp/4Q3/6B1/3qN3/8/PPP2PPP/R3R1K1 w kq - 1 16
- r3k2r/pppnb1pp/4Q3/6B1/3qN3/8/PPP2PPP/R3R1K1 w kq - 1 16
- r3k2r/pppnb1pp/4Q3/6B1/3qN3/8/PPP2PPP/R3R1K1 w kq - 1 16

RedPajama

- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nd7 11. Bg5 Qb4 12. Qg4 Qd4 13. Qxe6+ Be7 14. Qxe6+ Kf8 15. Qd6+ Ke8 16. Qd5+ Kf8 17. Qd6+
- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nd7 11. Bg5 Qb4 12. Qg4 Qd4 13. Qxe6+ Be7 14. Qxe6+ Kf8 15. f4 g6 16. Nf3 Qd7 17. Kb1 Kg7 18
- dxe4 Nxe4

Human Error Analysis

- **ChessGPT-Base**: Provides too much information, showing game records to go on, but not FEN.
- **ChessGPT-Chat**: The FEN provided is very similar to the requested answer, but with errors in the move counters.
- **RedPajama**: Completely wrong, it gives the game record instead of FEN.

Table 23: Question on PGN to FEN conversion.

UCI to FEN conversion.

In this task, the models were tasked to convert a chess game from UCI notation to FEN. Example responses can be found in table 24.

Prompt

Convert the following UCI to FEN: g2g3 d7d6 f1g2 e7e5 g1f3 g8f6 d2d4 b8 *(Answer: rnbqkb1r/ppp2ppp/3p1n2/4p3/3P4/5NP1/PPP1PPBP/RNBQK2R b KQkq - 0 4)*

ChessGPT-Base

- You can use the following code to convert the FEN string to a UCI string: #!/usr/bin/python import chess import chess.pygame import chess.uci import chess.pgn fen = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1' board = chess.Board(fen) board.parse_san(fen) # print(board.san(board.fen))
- The FEN is rnbqkb1r/ppp2ppp/3p1n2/4p3/3PP3/5NP1/PPP2PBP/RNBQK2R b KQkq - 0 5.
- Here is a link to the FEN representation of the above position: rnbqkb1r/ppp2ppp/3p1n2/4p3/3PP3/5NP1/PPP2PBP/RNBQK2R b KQkq - 0 5

ChessGPT-Chat

- rnbqkb1r/ppp2ppp/3p1n2/4p3/3P4/5NP1/PPP1PPBP/RNBQK2R w KQkq - 0 5
- rnbqkb1r/ppp2ppp/3p1n2/4p3/3P4/5NP1/PPP1PPBP/RNBQK2R w KQkq - 0 5
- rnbqkb1r/ppp2ppp/3p1n2/4p3/3P4/5NP1/PPP1PPBP/RNBQK2R b KQkq - 0 4

RedPajama

- Here is the FEN:g2k4f4 b8k4b5 d7k6d6 e7k6e5 f1k1f2 g8k8f6 UCI: g2k4f4 b8k4b5 d7k6d6 e7k6e5 f1k1f2 g8k8f6 Note that the FEN is a different notation than the UCI.
- Here is the FEN: g2g3 d7d6 f1g2 e7e5 g1f3 g8f6 d2d4 b8 The following are the chess pieces: g - knight g2 - bishop g3 - knight d - queen d7 - queen d6 - queen f - bishop f1 - bishop e - knight e7 - knight e5 - knight h - knight h2 - knight h3 - knight h4 - knight h5 - knight h6 - knight
- FEN: g2g3 d7d6 f1g2 e7e5 g1f3 g8f6 d2d4 b8 This is a standard position in chess, but not in FEN.

Human Error Analysis

- **ChessGPT-Base**:The first answer gives a code snippet rather than a direct answer. The second and third answers try to convert UCI to FEN, but the result is wrong. In terms of verbosity, too much information was provided, including an irrelevant code snippet.
- **ChessGPT-Chat**: The third answer is correct. The first two answers have subtle errors and the chess colors (w/b) are mislabeled.
- **RedPajama**: Completely wrong, it gives the game record instead of FEN.

Table 24: Question on UCI to FEN conversion.

PGN annotation

In this task, the models were tasked to generate commentary for a given game recorded in PGN format. Example responses can be found in table 25.

Prompt

1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Generate a step-by-step analysis

ChessGPT-Base

- This is a typical position in the Scotch Game. White has a lead in development and Black has a pawn weakness on e6. White's next move is 14. Bxe7.
- I believe you're looking for a program that will analyze the position and output the best move for both sides. I have a couple of suggestions: ChessBase: This is a commercial program, but it's free for the first 30 days. Scid vs. PC: This is free and open-source, but you need to download the source code. I have used both of these programs in the past.
- This is a pretty easy one. 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Rad1

ChessGPT-Chat

- 1. e4 e5 (Both players have opened with pawn moves 1. e4 and 1... e5, controlling the center and opening lines for their bishops and queen) 2. Nf3 d6 (Black has played pawn move 2... d6, which is a passive response to White's 2. Nf3) 3. d4 (White has played pawn move 3. d4, attacking Black's pawn on e5 and opening lines for the queen and bishop) exd4 (Black has captured White's pawn on d4 with his pawn on e5, surrendering the center to White) 4.
- 1. e4 e5 (Both players have opened with pawn moves to the e-file: 1. e4 and 1.... e5) 2. Nf3 d6 (Black has played the Philidor Defense, which is a solid, flexible response to 1. e4. Black aims to develop pieces and prepare for central pawn breaks) 3. d4 exd4 (Black captures the pawn on d4) 4. Nxd4 Nf6 (Black develops the knight and prepares to castle kingside) 5. Nc3 Qe7 (Black plays a queen move, attacking
- 15...Qxb2 (Black's best move) 16. Rab1 Qe5 (Black's only move to avoid losing material) 17. Qxe5 Nxe5 18. Rxb7 (White has a significant advantage due to Black's vulnerable king and misplaced knight on e5)

RedPajama

- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nxe4 11. Nxe4 Nd7 12. Bg5 Qb4 13. Qg4 Qd4 14. Qxe6+ Be7 15. Generate a step-by-step analysis
- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nd7 11. Bg5 Qb4 12. Qg4 Qd4 13. Qxe6+ Be7 14. Nd5 Nxd5 15. exd5 Nd7 16. c3 c6 17. Bf4 Qd8 18.
- 1. e4 e5 2. Nf3 d6 3. d4 exd4 4. Nxd4 Nf6 5. Nc3 Qe7 6. Bd3 d5 7. O-O dxe4 8. Re1 Be6 9. Nxe6 fxe6 10. Bxe4 Nd7 11. Bg5 Qb4 12. Qg4 Qd4 13. Qxe6+ Be7 14. Generate a step-by-step analysis

Table 25: Question on PGN annotation

Human Error Analysis

- **ChessGPT-Base**: The first answer is a short review of the Scotch Game, but not a step-by-step analysis. The second answer mentions other software for game analysis, which is not what the user asked for. The third answer just repeats the PGN given.
- **ChessGPT-Chat**: The first two answers started a step-by-step game analysis but were interrupted and not completed, but the analysis given is reasonable. The third answer provides a follow-up move for 15...Qxb2, but the user does not provide this step.
- **RedPajama**: The first and third answers just repeat the PGN given, no analysis is provided. The second answer tries to provide a follow-up move, which is also irrelevant to the question, since the user just requested an analysis, not a follow-up move.

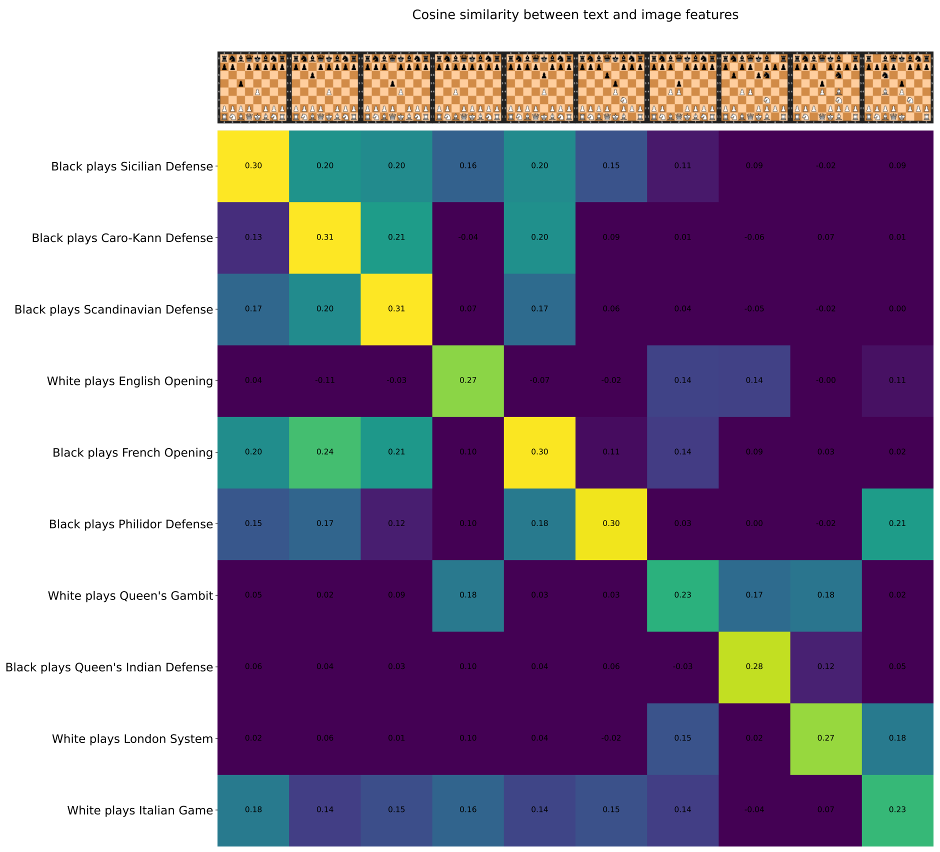


Figure 6: Similarity matrix of different chess opening PGN and text using ChessCLIP.

I ChessCLIP visualization

We present a visualization demo to better illustrate ChessCLIP's capability. Here we choose the chess opening as a test scenario and choose ten different chess openings. Ten chess opening PGNs and their corresponding names are arranged in sequence, so the diagonal cells in the entire similarity matrix should have the highest similarity. We present such a similarity matrix generated by ChessCLIP in Figure 6. The results exactly correspond to our expectations, which successfully validate the effectiveness of our ChessCLIP model.

J Potential directions

In this section we will describe several potential directions based on our dataset.

1. Dataset Augmentation and Fine-tuning: Researchers can explore dataset augmentation techniques to further enhance the diversity and size of the provided dataset. By introducing variations in game conditions, player strategies, or opening positions, researchers can create augmented datasets that can be used to fine-tune existing language models or train new models specifically tailored for chess-related tasks. This can potentially lead to improved performance in state tracking, value judgement, and policy evaluation.
2. Transfer Learning to Chess Variants: The dataset and benchmark provided in this study can serve as a valuable resource for transfer learning experiments to chess variants. Researchers can leverage the knowledge learned from the base chess task and apply it to chess variants such as Fischer Random Chess or Chess960. By fine-tuning or adapting pre-trained language models on the provided dataset, researchers can explore the capabilities of these models in handling different chess variants, including state tracking, value judgement, and policy evaluation
3. Reinforcement Learning/Planning with Language Models: Combining reinforcement learning or planning with language models trained on the provided dataset opens up exciting possibilities for improving chess-playing agents. Researchers can develop reinforcement learning algorithms that

utilize the language model's state tracking ability to build more sophisticated and strategic agents. By training agents to interact with the language model in a dialogue-like manner, researchers can explore the potential of language-based reinforcement learning for chess-related tasks, such as move generation and evaluation..

4. Explainable AI in Chess: Given the language model's ability to generate human-readable outputs, researchers can investigate the application of explainable AI techniques in chess. By interpreting the model's generated moves or predictions, researchers can gain insights into the reasoning behind the model's decisions. This can lead to the development of explainable AI systems that provide justifications or explanations for their chess moves, aiding both players and analysts in understanding and learning from the model's decision-making process.
5. Multi-modal Approaches: Researchers can explore multi-modal approaches that combine textual and visual information for chess-related tasks. By incorporating board visualizations, game position images, or move sequence visualizations along with textual inputs, researchers can develop models that leverage both textual and visual cues to improve state tracking, value judgement, and policy evaluation in chess. This can open up avenues for multi-modal analysis and understanding of chess games, allowing models to capture and reason over both textual and visual representations simultaneously.
6. Chess Education and Tutorial Systems: The dataset can be utilized to develop educational tools and tutorial systems for chess players of different skill levels. Researchers can leverage the language model's expertise in state tracking, value judgement, and policy evaluation to provide interactive and personalized learning experiences. By tailoring the tutorial content and feedback based on individual player performance, researchers can create intelligent systems that assist in skill development and strategic improvement in chess.
7. Adversarial Attacks in Chess: With the increasing use of language models in critical applications like chess analysis and decision-making, it becomes essential to investigate potential vulnerabilities and develop defenses against adversarial attacks. Researchers can explore techniques to generate adversarial examples specifically targeted at chess-related tasks. By identifying weaknesses in language models' state tracking or policy evaluation abilities, researchers can enhance the robustness and security of these models.
8. Chess Game Generation: Researchers can utilize the provided dataset to develop models capable of generating new chess game sequences. By leveraging the language model's understanding of chess moves and game structures, researchers can explore generative models that can produce novel and diverse chess game sequences. This can be beneficial for various applications, including chess game analysis, training data generation, and even game generation for chess variants.

Overall, our dataset and benchmark offer numerous potential directions, ranging from dataset expansion and transfer learning to exploring chess variants, education, analysis, and game generation. These directions have the potential to advance the field of chess-related language modeling and provide valuable tools and resources for chess players and enthusiasts.

K Limitations and Potential Societal Impact

The availability of a comprehensive and diverse chess dataset presents both limitations and potential societal impacts that researchers should consider.

Limitations. While the chess dataset provided in this study is valuable, it is important to acknowledge its limitations. One limitation is the potential bias introduced by relying on historical Lichess matches from different time periods. This may result in variations in player strategies, popular openings, and game trends over time, potentially impacting the generalizability of the dataset. Additionally, it is worth noting that the dataset predominantly focuses on standard chess, and may not encompass the full spectrum of chess variants. Researchers interested in exploring niche or less-popular variants may need to gather additional data from specific sources to ensure comprehensive coverage of different chess variants. These considerations are crucial to ensure the validity and applicability of research findings based on the provided dataset.

Potential Societal Impact. The availability of a comprehensive and diverse chess dataset can have a significant societal impact. First and foremost, it can contribute to the development of more advanced and intelligent chess-playing agents. These agents can be utilized in various applications, such as chess analysis, training tools for players of different skill levels, and even as opponents for chess enthusiasts. The dataset can also facilitate the advancement of chess education by providing valuable resources for tutorials, interactive learning platforms, and strategic guidance. Additionally, the dataset can inspire research in the field of artificial intelligence, contributing to the development of innovative techniques that can be applied beyond the domain of chess. Lastly, the dataset can encourage the exploration of explainable AI methods in chess, enabling players to understand and learn from the reasoning behind the model's moves, thereby promoting transparency and trust in AI systems.