Automated Chess Commentator Powered by Neural Chess Engine

Hongyu Zang* and **Zhiwei Yu*** and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{zanghy, yuzw, wanxiaojun}@pku.edu.cn

Abstract

In this paper, we explore a new approach for automated chess commentary generation, which aims to generate chess commentary texts in different categories (e.g., *description*, *comparison*, *planning*, etc.). We introduce a neural chess engine into text generation models to help with encoding boards, predicting moves, and analyzing situations. By jointly training the neural chess engine and the generation models for different categories, the models become more effective. We conduct experiments on 5 categories in a benchmark Chess Commentary dataset and achieve inspiring results in both automatic and human evaluations.

1 Introduction

With games exploding in popularity, the demand for Natural Language Generation (NLG) applications for games is growing rapidly. Related researches about generating real-time game reports (Yao et al., 2017), comments (Jhamtani et al., 2018; Kameko et al., 2015), and tutorials (Green et al., 2018a,b) benefit people with entertainments and learning materials. Among these, chess commentary is a typical task. As illustrated in Figure 1, the commentators need to understand the current board and move. And then they comment about the current move (*Description*), their judgment about the move (*Quality*), the game situation for both sides (*Contexts*), their analysis (*Comparison*) and guesses about player's strategy (*Planning*). The comments provide valuable information about what is going on and what will happen. Such information not only make the game more enjoyable for the viewers, but also help them learn to think and play. Our task is to design automated generation model to address all the 5 sub-tasks (*Description*, *Quality*, *Comparison*, *Planning*, and *Contexts*) of single-move chess commentary.

*The two authors contributed equally to this paper.

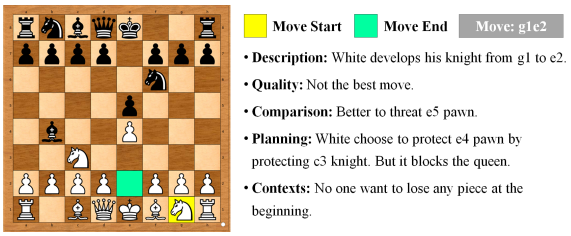


Figure 1: Chess Commentary Examples.

Automatically generating chess comments draws attention from researchers for a long time. Traditional template-based methods (Sadikov et al., 2007) are precise but limited in template variety. With the development of deep learning, data-driven methods using neural networks are proposed to produce comments with high quality and flexibility. However, generating insightful comments (e.g., to explain why a move is better than the others) is still very challenging. Current neural approaches (Kameko et al., 2015; Jhamtani et al., 2018) get semantic representations from raw boards, moves, and evaluation information (threats and scores) from external chess engines. Such methods can easily ground comments to current boards and moves. But they cannot provide sufficient analysis on what will happen next in the game. Although external features are provided by powerful chess engines, the features are not in a continuous space, which may be not very suitable for context modeling and commentary generation.

It is common knowledge that professional game commentators are usually game players. And expert players can usually provide more thorough analysis than amateurs. Inspired by this, we argue that for chess commentary generation, the generation model needs to know how to think and play in order to provide better outputs. In this paper, we introduce a neural chess engine into our generation models. The chess engine is

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5952–5961
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

pre-trained by supervised expert games collected from FICS Database[1] and unsupervised self-play (Silver et al., 2017a,b) games, and then jointly trained with the generation models. It is able to get board representations, predict reasonable move distributions, and give continuous predictions by self-play. Our generation models are designed to imitate commentators' thinking process by using the representations and predictions from the internal chess engine. And then the models ground commentary texts to the thinking results (semantics). We perform our experiments on 5 categories (*Description*, *Quality*, *Contexts*, *Comparison*, *Planning*) in the benchmark Chess Commentary dataset provided by Harsh (2018). We tried models with different chess engines having different playing strength. Both automatic and human evaluation results show the efficacy and superiority of our proposed models.

The contributions are summarized as follows:

- To the best of our knowledge, we are the first to introduce a compatible neural chess engine to the chess comment generation models and jointly train them, which enables the generation models benefit a lot from internal representations of game playing and analysis.
- On all the 5 categories in the Chess Commentary dataset, our proposed model performs significantly better than previous state-of-the-art models.
- Our codes for models and data processing will be released on GitHub[2]. Experiments can be easily reproduced and extended.

2 Related Works

The most relevant work is (Jhamtani et al., 2018). The authors released the Chess Commentary dataset with the state-of-the-art Game Aware Commentary (GAC) generation models. Their models generate comments with extracted features from powerful search-based chess engines. We follow their work to further explore better solutions on different sub-tasks (categories) in their dataset. Another relevant research about Shogi (a similar board game to chess) commentary generation is from Kameko et al. (2015). They rely on external tools to extract key words first, and then generate comments with respect to the key words. Different from their works, in this paper, we argue that an internal neural chess engine can provide better information about the game states, options and developments. And we design reasonable models and sufficient experiments to support our proposal.

Chess engine has been researched for decades (Levy and Newborn, 1982; Baxter et al., 2000; David et al., 2017; Silver et al., 2017a). Powerful chess engines have already achieved much better game strength than human-beings (Campbell et al., 2002; Silver et al., 2017a). Traditional chess engines are based on rules and heuristic searches (Marsland, 1987; Campbell et al., 2002). They are powerful, but limited to the human-designed value functions. In recent years, neural models (Silver et al., 2016, 2017b; David et al., 2017) show their unlimited potential in board games. Several models are proposed and can easily beat the best human players in Go, Chess, Shogi, etc. (Silver et al., 2017a). Compared to the traditional engines, the hidden states of neural engines can provide vast information about the game and have the potential to be compatible in NLG models. We follow the advanced techniques and design our neural chess engine. Apart from learning to play the game, our engine is designed to make game states compatible with semantic representations, which bridges the game state space and human language space. And to realize this, we deploy multi-task learning (Collobert and Weston, 2008; Sanh et al., 2018) in our proposed models.

Data-to-text generation is a popular track in NLG researches. Recent researches are mainly about generating from structured data to biography (Sha et al., 2018), market comments (Murakami et al., 2017), and game reports (Li and Wan, 2018). Here we manage to ground the commentary to the game data (boards and moves). Addressing content selection (Wiseman et al., 2017) is one of the top considerations in our designs.

3 Our Approach

The overview of our approach is shown in Figure 2. Apart from the text generation models, there are three crucial modules in our approach: the internal chess engine, the move encoder, and the multi-choices encoder. We will first introduce our solution to all the sub-tasks of chess commentary generation with the modules as black boxes. And then

[1] https://www.ficsgames.org/
[2] https://github.com/zhyack/SCC

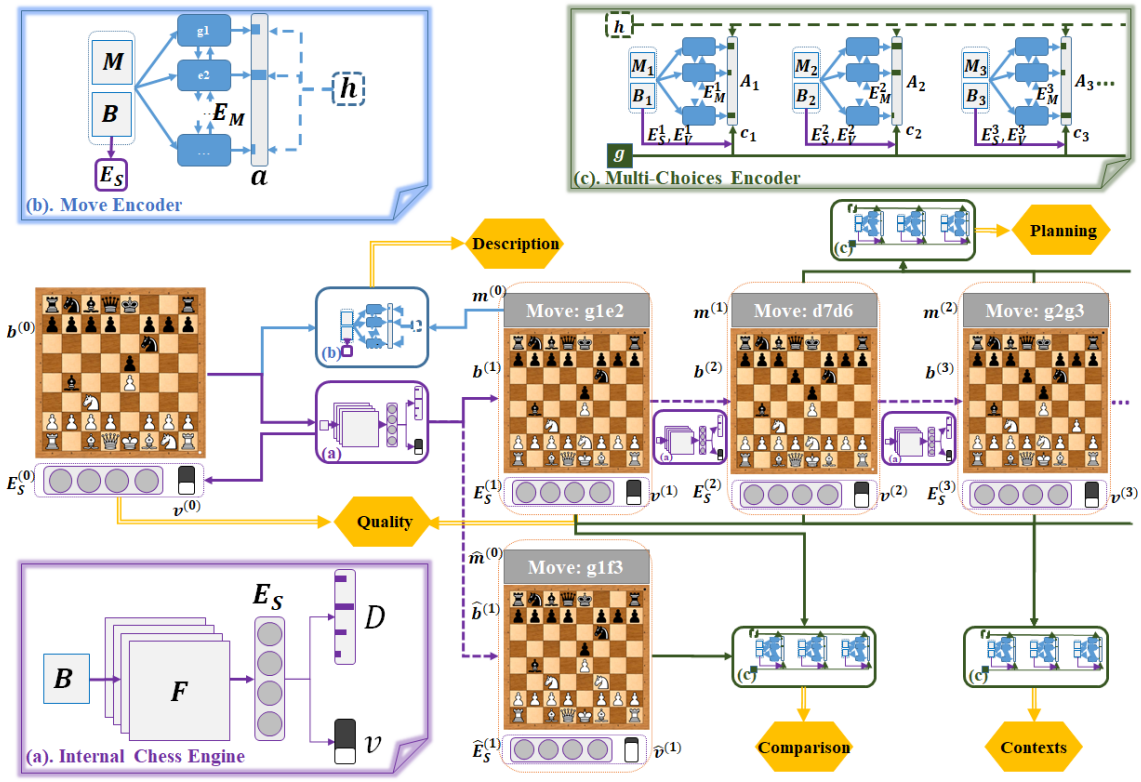


Figure 2: Overview of our chess commentary model.

we describe them in details.

3.1 Our Solutions

In Figure 2, an example is presented with model structures to demonstrate the way our models solving all the sub-tasks. The process is impelled by the internal chess engine. Given the current board $b^{(0)}$ and move $m^{(0)}$, the engine emulates the game and provides the current and next board states together with wining rates of the players. Besides, the engine also predicts for another optional move $\hat{m}^{(0)}$ from $b^{(0)}$ to make comparisons to $m^{(0)}$. And then a series of long-term moves $(m^{(1)}, m^{(2)}, ...)$ and boards $(b^{(2)}, b^{(3)}, ...)$ are further predicted by the engine in a self-play manner (Silver et al., 2017a,b) for deep analysis. With the semantics provided by the engine, generation models are able to predict with abundant and informative contexts. We will first detail the different semantic contexts with respect to models for 5 different subtasks. And then we summarize the common decoding process for all the models.

Description Model: Descriptions about the current move intuitively depend on the move itself. However, playing the same move could have different motivations under different contexts. For example, *e2e4* is the classic Queen Pawn Opening in a fresh start. But it can be forming a pawn defense structure in the middle of the game. Different from previous works for chess commentary generation (Jhamtani et al., 2018; Kameko et al., 2015), we find all kinds of latent relationships in the current board vital for current move analysis. Therefore, our description model takes the representation of both $b^{(0)}$ and $m^{(0)}$ from the move encoder f_{ME} as semantic contexts to produce description comment Y_{Desc}. The description model is formulated as Eq.1.

$$f_{Description}(f_{ME}(b^{(0)}, m^{(0)})) \rightarrow Y_{Desc} \quad (1)$$

Quality Model: Harsh et al. (2018) find the wining rate features benefit the generation models on *Quality* category. Inspired by this, we concatenate the current board state $E_S^{(0)}$, the next board state $E_S^{(1)}$, and the wining rate difference $v^{(1)} - v^{(0)}$ as semantic contexts for the decoder. And to model the value of wining rate difference, we introduce a weight matrix W_{diff} to map the board state-value pair $[E_S^{(0)}; E_S^{(1)}; v^{(1)} - v^{(0)}]$ to the same semantic space of the other contexts by Eq.2. Our quality model is formulated as Eq.3, where Y_{Qual} is the target comment about quality.

$$E_D = W_{diff}[E_S^{(0)}; E_S^{(1)}; v^{(1)} - v^{(0)}] \quad (2)$$

$$f_{Quality}(E_S^{(0)}, E_S^{(1)}, E_D) \rightarrow Y_{Qual} \quad (3)$$

Comparison Model: Usually, there are more than 10 possible moves in a given board. But not all of them are worth considering. Kameko et al. (2015) propose an interesting phenomenon in chess commentary: when the expert commentators comment about a bad move, they usually explain why the move is bad by showing the right move, but not another bad move. Inspired by this, we only consider the true move $m^{(0)}$ and the potential best move $\hat{m}^{(0)}$ (decided by the internal chess engine) as options for the comparison model. And the semantic contexts for the options are encoded by the multi-choices encoder. We define the comparison model as Eq.4 , where f_{MCE} is the multi-choices encoder, $b^{(1)}$ is the board after executing $m^{(0)}$ on $b^{(0)}$, $\hat{b}^{(1)}$ is the board after executing $\hat{m}^{(0)}$ on $b^{(0)}$, and Y_{Comp} is the target comment about comparison.

$$f_{Comparison}(f_{MCE}((b^{(1)}, m^{(0)}), (\hat{b}^{(1)}, \hat{m}^{(0)}))) \rightarrow Y_{Comp} \quad (4)$$

Planning Model: We can always find such scenes where commentators try to predict what will happen assuming they are playing the game. And then they give analysis according to their simulations. Our internal chess engine is able to simulate and predict the game in a similar way (self-play). We realize our model for planning by imitating the human commentators' behavior. Predicted moves and boards are processed by our multi-choices encoder to tell the potential big moments in the future. And we use the multi-choices encoder f_{MCE} to produce the semantic contexts for the decoder. The process to generate planning comment Y_{Plan} is described in Eq.5.

$$f_{Planning}(f_{MCE}((b^{(2)}, m^{(1)}), (b^{(3)}, m^{(2)}), (b^{(4)}, m^{(3)}), ...)) \rightarrow Y_{Plan} \quad (5)$$

Contexts Model: To analyze the situation of the whole game, the model should know about not only the current, but also the future. And similar to the planning model, contexts model takes a series of long-term moves and boards produced by self-play predictions as inputs. In this way, the model comments the game in a god-like perspective. And the semantic contexts is also processed by the multi-choices encoder for generating contexts comment Y_{Cont} as Eq.6.

$$f_{Contexts}(f_{MCE}((b^{(1)}, m^{(0)}), (b^{(2)}, m^{(1)}), (b^{(3)}, m^{(2)}), (b^{(4)}, m^{(3)}), ...)) \rightarrow Y_{Cont} \quad (6)$$

Each of the above models has a decoder (the hexagon blocks in Figure 2) for text generation and we use LSTM decoders (Sundermeyer et al., 2012). And we use cross entropy loss function for training. The function is formalized as Eq.7, where Y is the gold standard outputs.

$$Loss_{Gen} = -logp(Y|b^{(0)}; m^{(0)}) \quad (7)$$

We denote $E \in \mathbb{R}^{n \times d}$ as a bunch of raw context vectors, where n is the number of such context vectors and d is the dimension of the vectors. Although the semantic contexts E for different generation models are different as described before, we regard all of the board states, wining rates, and move representations as general semantic contexts. And we use attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) to gather information from the contexts. For example, assuming that we have a hidden vector h drawing from LSTM units, to decode with the semantic contexts, we use the score function f of Luong attention (Luong et al., 2015) as

$$f(X, y) = XWy, \quad (8)$$

to calculate the attention weights a for vectors in E, where W is a transformation function for the attentional context vectors. The scores are further normalized by a softmax function to a by

$$a = \boldsymbol{softmax}(f(E, h)). \quad (9)$$

We compute weighted sum of E with a to produce the attentional context vector z for word decoding

$$z = E^{\top} a. \quad (10)$$

3.2 The Internal Chess Engine

The internal chess engine is in charge of the mapping from board B to semantic representation E_S, predicting possibility distribution D on valid moves, and evaluating the wining rate v for the players. In previous works (Jhamtani et al., 2018; Kameko et al., 2015), researchers use discrete information (threats, game evaluation scores, etc.) analyzed by external chess engine to build semantic representations. It limits the capability of the

representations by simply mapping the independent features. Our internal chess engine is able to mine deeper relations and semantics with the raw board as input. And it can also make predictions in a continuous semantic space, increasing the capability and robustness for generation.

Following advanced researches in neural chess engines (David et al., 2017; Silver et al., 2017a), we split the input raw board into 20 feature planes F for the sake of machine understanding. There are 12 planes for pieces' (pawn, rook, knight, bishop, queen, king) positions of each player, 4 planes for white's repetitions, black's repetitions, total moves, and moves with no progress, and 4 planes for 2 castling choices of each player. The feature planes F are encoded by several CNN layers to produce sufficient information for semantic representation E_S. Like previous researches on chess engines, E_S is used to predict the move possibility distribution D and the wining rate v by fully connected layers. But different from those pure engines, we share the board state E_S with generation models in a multi-task manner (Collobert and Weston, 2008). The engine is designed not only for playing, but also for expressing. Our generation models use E_S as part of the inputs to get better understanding of the game states.

Given the tuple of game replays (B, M, v') where M is the corresponding move and v' is the ground truth wining rate, we optimize the engine's policy, value function at the same time as Eq.11 shows. When the engine grows stronger, we let the engine produce data by itself in a self-play manner (Silver et al., 2017a). Besides, the engine jointly optimizes $Loss_{Gen}$ when training generative models.

$$Loss_{Eng} = -logp(M|B) + (v - v')^2 \quad (11)$$

3.3 The Move Encoder

Apart from understanding the board B, commentators also need to know the semantics of the move M. Besides using the chess engine to produce board representations E_S, the move encoders also prepare for move embeddings E_M as attention contexts for the text decoders. We set the features of the move (starting cell, the move ending cell, the piece at the starting cell, the piece at the ending cell, the promotion state, and the checking state) as a sequential input to a bi-directional RNN (Schuster and Paliwal, 1997). When a decoder requests attention contexts for hidden state h, the encoder offers $E = [E_M; E_S]$ to build attentional context vector following Eq.9 and Eq.10.

3.4 The Multi-Choices Encoder

For *Comparison*, *Planning*, and *Contexts*, there are multiple moves derived from variations and predictions. The model needs to find the bright spots to describe. To encode these moves and offer precise information for the generation models, we propose a multi-choices encoder. Human commentators usually choose different aspects to comment according to their experiences. We use a global vector g to store our models' experiences and choose important moves to comment. Note that g is to be learned. In module (c) of Figure 2, we denote E^i_M as the output vectors of the i-th move encoder, E^i_S as the board state of the i-th board, and E^i_V as the embedding of wining rate v^i of the i-th board. To model the wining rate value, we introduce a mapping matrix M_{val} and process the state-value pair to the value embedding as

$$E^i_V = W_{val}[E^i_S, v^i]. \quad (12)$$

Then we calculate the soft weights of choices $c = \{c_1, c_2, ...\}$ with respect to the board states $S = \{E^1_S, E^2_S, ...\}$ by Eq.13. For hidden state vector h from decoder, attention weight matrix $A = \{A_1, A_2, ...\}$ are scaled by c via Eq.14. And we finally get attentional context vector z according to A by Eq.15. This approach enables generation models to generate comments with attention to intriguing board states. And the attention weights can be more accurate when g accumulates abundant experiences in training.

$$c = \boldsymbol{softmax}(gS) \quad (13)$$

$$A_i = c_i * \boldsymbol{softmax}(f([E^i_M; E^i_S; E^i_V], h)) \quad (14)$$

$$z = \sum_i ([E^i_M; E^i_S; E^i_V])^\top A_i \quad (15)$$

4 Experiments

4.1 Dataset

We conduct our experiments on recently proposed Chess Commentary dataset[3] (Jhamtani et al., 2018). In this dataset, Harsh et al. (2018) collect and process 11,578 annotated chess games from a large social forum GAMEKNOT[4]. There are 298K aligned data pairs of game moves and

[3]https://github.com/harsh19/ChessCommentaryGeneration/
[4]https://gameknot.com

commentaries. The dataset is split into training set, validation set and test set as a 7:1:2 ratio with respect to the games. As the GAMEKNOT is a free-speech forum, the comments can be very free-wheeling in grammar and morphology. The informal language style and unpredictable expression tendency make a big challenge for data-driven neural generation models. To narrow down the expression tendency, Harsh et al. (2018) classify the dataset into 6 categories: *Description*, *Quality*, *Comparison*, *Planning*, *Contexts*, and *General*. The *General* category is usually about the player and tournament information, which needs external knowledge irrelevant to game analysis. We do not conduct experiments on the last category.

And for the training of chess engine, we collect all of the standard chess game records in the past 10 years from FICS Games Database. And we remove the games where any player's rating below 2,000. There are 36M training data (for single move step) after cleaning.

4.2 Experiment Settings and Baselines

We train our neural chess engine using mixed data consisting of supervised FICS data and unsupervised self-play data. The number of self-play games are set to 0 initially. And it will be increased by 1 when the trained model beats the previous best version (with a wining rate larger than 0.55 in 20 games). During 400 iterations of training, we pick one strong engine and one weak engine for further experiments. The stronger engine loses 1 game and draws 55 games to the weak engine in 100 games. As mentioned in Section 3.2, when training generation models, we use the pre-trained chess engine and fine-tune it with the generation models.

Here we introduce our models and baselines in the experiments. We call our models the Skilled Chess Commentator (SCC) as they have the skills of playing chess.

- **SCC-weak**: The generation models are integrated with the weak engine mentioned above, and they are trained independently with respect to the 5 categories in Chess Commentary dataset.
- **SCC-strong**: The model is similar to SCC-weak, but integrated with the strong engine.
- **SCC-mult**: This is a multi-task learning model where generation models for different categories share the strong chess engine, move encoder, the multi-choices encoder and the value mapping matrix W_{val}.
- **GAC**: The state-of-the-art method proposed by Harsh et al. (2018). Their models incorporate the domain knowledge provided by external chess engines. Their models only work for first 3 categories: *Description*, *Quality*, and *Comparison*. We will compare our results with **GAC** on these categories.
- **KWG**: Another state-of-the-art method for game commentary generation (Kameko et al., 2015). It is a pipeline method based on keyword generation. We compare the results on all data categories.
- **Temp**: This is a template-based baseline methods. Together with the dataset, Harsh et al. (2018) provide templates for the first two categories. Inspired by (Sadikov et al., 2006), we extend the templates to fit for all the 5 categories.
- **Re**: This is a retrieval-based baseline method. For each input in the test set, we find the most matched datum in the training set by numbers of matched input board and move features.

4.3 Evaluation Metrics

We develop both automatic evaluations and human evaluations to compare the models.

For automatic evaluations, we use BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) to evaluate the generated comments with ground-truth outputs. BLEU evaluates the modified precision between the predicted texts and gold-standard references on corpus level. Evaluating with 4-grams (BLEU-4 [5]) is the most popular way in NLG researches. However, for tasks like dialogue system (Li et al., 2016), story telling generation (Jain et al., 2017), and chess commentary (Jhamtani et al., 2018), the outputs can be rather short and free expressions. Under such circumstances, brevity penalty for 4-grams can be too strict and makes the results unbalanced. We use BLEU-2 [6] to show more steady results with BLEU

[5] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

[6] https://github.com/harsh19/ChessCommentaryGeneration/blob/master/Code/methods/category_aware/BLEU2.perl

BLEU-4 (%)	Temp	Re	KWG	GAC	SCC-weak	SCC-strong	SCC-mult
Description	0.82	1.24	1.22	**1.42**	1.23	1.31	1.34
Quality	13.71	4.91	13.62	16.90	16.83	18.87	**20.06**
Comparison	0.11	1.03	1.07	1.37	2.33	**3.05**	2.53
Planning	0.05	0.57	0.84	N/A	**1.07**	0.99	0.90
Contexts	1.94	2.70	4.39	N/A	4.04	**6.21**	4.09
BLEU-2 (%)	**Temp**	**Re**	**KWG**	**GAC**	**SCC-weak**	**SCC-strong**	**SCC-mult**
Description	24.42	22.11	18.69	19.46	23.29	**25.98**	25.87
Quality	46.29	39.14	55.13	47.80	58.53	61.13	**61.62**
Comparison	7.33	22.58	20.06	24.89	24.85	**27.48**	23.47
Planning	3.38	20.34	22.02	N/A	22.28	**25.82**	24.32
Contexts	26.03	30.12	31.58	N/A	37.32	**41.59**	38.59
METEOR (%)	**Temp**	**Re**	**KWG**	**GAC**	**SCC-weak**	**SCC-strong**	**SCC-mult**
Description	6.26	5.27	6.07	6.19	6.03	6.83	**7.10**
Quality	22.95	17.01	22.86	24.20	24.89	**25.57**	25.37
Comparison	4.27	8.00	7.70	8.54	8.25	**9.44**	9.13
Planning	3.05	6.00	6.76	N/A	6.18	7.14	**7.30**
Contexts	9.46	8.90	10.31	N/A	11.07	**11.76**	11.09

Table 1: Automatic evaluation results.

evaluation algorithm. We also use METEOR as a metric, whose results are more closed to a normal distribution (Dobre, 2015).

We also conduct human evaluation to make more convincing comparisons. We recruit 10 workers on Amazon Mechanical Turk[7] to evaluate 150 groups of samples (30 from each category). Each sample is assigned to exactly 2 workers. The workers rate 8 shuffled texts (for **Ground Truth**, **Temp**, **Re**, **GAC**, **KWG**, and **SCC** models) for the following 4 aspect in a 5-pt Likert scale[8].

- **Fluency**: Whether the comment is fluent and grammatical.
- **Accuracy**: Whether the comment correctly describes current board and move.
- **Insights**: Whether the comment makes appropriate predictions and thorough analysis.
- **Overall**: The annotators' overall impression about comments.

4.4 Results and Analysis

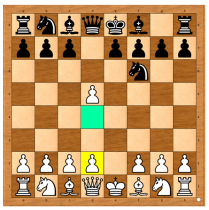
We present the automatic evaluation results in Table 1. Our **SCC** models outperform all of the baselines and previous state-of-the-art models. **Temp** is limited by the variety of templates. It is competitive with the neural models on *Description* and *Quality* due to limited expressions in these tasks. But when coming to *Comparison*, *Planning* and *Contexts*, **Temp** shows really bad performances. **Re** keeps flexibility by copying the sentences from training set. But it does not perform well, either. The ability of **Re** is limited by the sparse searching space, where there are 90,743 data in the training set, but 10^{43} possible boards[9] for chess game. **KWG** and **GAC** provide competitive results. With the help of external information from powerful chess engines, **GAC** shows good performances on *Quality* and *Comparison*. Although our internal chess engine is no match for the external engines that **GAC** uses at playing chess, it turns out that our models with directly internal information can better bridge the semantic spaces of chess game and comment language. As for the comparisons within our models, **SCC-strong** turns to be better than **SCC-weak**, which supports our assumption that better skills enable more precise predictions, resulting in better comments. Training with multi-task learning seems to hurt the overall performances a little. But **SCC-mult** still has the state-of-the-art performances. And more important, it can react to all sub-tasks as a whole.

The human annotators are required to be good

[7] https://www.mturk.com
[8] https://en.wikipedia.org/wiki/Likert scale
[9] https://en.wikipedia.org/wiki/Shannon_number

Move Start Move End Move: g1e2 Description

- **Ground Truth:** Maintains protection of the King's pawn by protecting the threatened knight with the other knight .
- **Temp:** White moves the knight from g1 to e2 .
- **Re:** I defend and develop with my other knight .
- **KWG:** White develops his knight to e7 and attacks my knight .
- **GAC:** White develops his knight .
- **Our Model:** White develops his knight to protect the pawn and protects the knight .



Move Start Move End Move: d2d4 Planning

- **Ground Truth:** Here , white gives the pawn back and continues his development , also opening lines for the c1 Bishop .
- **Temp:** White tries to win the game little by little .
- **Re:** Standard queen 's pawn opening .
- **KWG:** White open with his queen pawn .
- **GAC:** N/A
- **Our Model:** White 's pawn is opening up the bishop and queen on the diagonal .

Figure 3: Samples for case study.

Models	Fluency	Accuracy	Insights	Overall
Ground Truth	**4.02**	**3.88**	**3.58**	**3.84**
Temp	**4.05**	**4.03**	3.02	3.56
Re	3.71	3.00	2.80	2.85
KWG	3.51	3.24	2.93	3.00
SCC-weak	3.63	3.62	3.32	3.30
SCC-strong	3.81	3.74	3.49	3.49
SCC-mult	3.82	3.91	**3.51**	**3.61**
GAC*	3.68	3.32	2.99	3.14
SCC-mult*	3.83	3.99	3.46	3.52

Table 2: Human evaluation results. Models marked with * are evaluated only for the *Description*, *Quality*, and *Comparison* categories. The underlined results are significantly worse than those of **SCC-mult(*)** in a two-tail T-test ($p<0.01$).

at playing chess. That is to say, they are the true audiences of the commentator researches and applications. By introducing human evaluations, we further reveal the performances in the perspective of the audiences. We show the average scores and significance test results in Table 2. We further demonstrate the efficacy of our models with significantly better overall performances than the retrieval-based model and previous state-of-the-art ones. It is worth noting that the evaluations about Accuracy and Insights show that our models can produce more precise and thorough analysis owing to the internal chess engine. **SCC-mult** and **SCC-strong** perform better than **SCC-weak** in Accuracy and Overall scores. It also supports the points that the our commentary model can be improved with better internal engine.

4.5 Case Study

To have a better view of comparisons among model outputs, we present and analyze some samples in Figure 3. In these samples, our model refers to **SCC-mult**.

For the first example, black can exchange white's *e3* knight and *e4* pawn with the *b4* bishop if white takes no action. But white chooses to protect the *e3* knight with the *g1* knight. All the models generate comments about *Description*. **Temp** directly describes the move without explanation. **Re** finds similar situation in the training set and explains the move as defense and developing. **KWG** is right about developing, but wrong about the position of the knight and the threats. **GAC** produces safe comment about the developing. And our model has a better understanding about the boards. It annotates the move correctly and even gives the reason why white plays this move.

For the second example, the game is at the 3rd turn. White gives up the pawn on *d5* and chooses to push the queen's pawn. **Re** and **KWG** both make a mistake and recognize the move *d2d4* as Queen Pawn Opening. **Temp** thinks white is going to win because white have the advantage of one more pawn. However, **Temp** cannot predict that white will lose the advantage in the next move. Our model is able to predict the future moves via self-play. And it draws the conclusion that pushing the queen's pawn can open up the ways for the queen and bishop for future planning.

5 Conclusion and Future Work

In this work we propose a new approach for automated chess commentary generation. We come up with the idea that models capable of playing chess will generate good comments, and models with better playing strength will perform better in generation. By introducing a compatible chess engine to comment generation models, we get models that can mine deeper information and ground

more insightful comments to the input boards and moves. Comprehensive experiments demonstrate the effectiveness of our models.

Our experiment results show the direction to further developing the state-of-the-art chess engine to improve generation models. Another interesting direction is to extend our models to multi-move commentary generation tasks. And unsupervised approaches to leverage massive chess comments in social media is also worth exploring.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jonathan Baxter, Andrew Tridgell, and Lex Weaver. 2000. Learning to play chess using temporal differences. *Machine Learning*, 40(3):243–263.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. 2002. Deep blue. *Artificial intelligence*, 134(1-2):57–83.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of (ICML 2008)*, pages 160–167.

Eli David, Nathan S. Netanyahu, and Lior Wolf. 2017. Deepchess: End-to-end deep neural network for automatic learning in chess. *CoRR*, abs/1711.09667.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Iuliana Dobre. 2015. A comparison between bleu and meteor metrics used for assessing students within an informatics discipline course. *Procedia-Social and Behavioral Sciences*, 180:305–312.

Michael Cerny Green, Ahmed Khalifa, Gabriella A. B. Barros, Andy Nealen, and Julian Togelius. 2018a. Generating levels that teach mechanics. In *Proceedings of the 13th International Conference on the Foundations of Digital Games, FDG 2018, Malmö, Sweden, August 07-10, 2018*, pages 55:1–55:8.

Michael Cerny Green, Ahmed Khalifa, Gabriella A. B. Barros, and Julian Togelius. 2018b. "press space to fire": Automatic video game tutorial generation. *CoRR*, abs/1805.11768.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *CoRR*, abs/1707.05501.

Harsh Jhamtani, Varun Gangal, Eduard H. Hovy, Graham Neubig, and Taylor Berg-Kirkpatrick. 2018. Learning to generate move-by-move commentary for chess games from large-scale social forum data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1661–1671.

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Learning a game commentary generator with grounded move expressions. In *2015 IEEE Conference on Computational Intelligence and Games, CIG 2015, Tainan, Taiwan, August 31 - September 2, 2015*, pages 177–184.

David Levy and Monroe Newborn. 1982. How computers play chess. In *All About Chess and Computers*, pages 24–39. Springer.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Liunian Li and Xiaojun Wan. 2018. Point precisely: Towards ensuring the precision of data in generated texts using delayed copy mechanism. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1044–1055.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*, pages 1412–1421.

Tony A Marsland. 1987. Computer chess methods. *Encyclopedia of Artificial Intelligence*, 1:159–171.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the*

Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 1374–1384.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Aleksander Sadikov, Martin Movzina, Matej Guid, Jana Krivec, and Ivan Bratko. 2007. Automated chess tutor. In *Computers and Games*, pages 13–25, Berlin, Heidelberg. Springer Berlin Heidelberg.

Aleksander Sadikov, Martin Mozina, Matej Guid, Jana Krivec, and Ivan Bratko. 2006. Automated chess tutor. In *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, pages 13–25.

Victor Sanh, Thomas Wolf, and Sebastian Ruder. 2018. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *AAAI 2019*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5414–5421.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017a. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017b. Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Jin-ge Yao, Jianmin Zhang, Xiaojun Wan, and Jianguo Xiao. 2017. Content selection for real-time sports news construction from commentary texts. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 31–40.