Learning to Generate Move-by-Move Commentary for Chess Games from Large-Scale Social Forum Data

Harsh Jhamtani[*], **Varun Gangal**[*], **Eduard Hovy, Graham Neubig, Taylor Berg-Kirkpatrick**
Language Technologies Institute
Carnegie Mellon University
{jharsh,vgangal,hovy,gneubig,tberg}@cs.cmu.edu

Abstract

This paper examines the problem of generating natural language descriptions of chess games. We introduce a new large-scale chess commentary dataset and propose methods to generate commentary for individual moves in a chess game. The introduced dataset consists of more than 298K chess move-commentary pairs across 11K chess games. We highlight how this task poses unique research challenges in natural language generation: the data contain a large variety of styles of commentary and frequently depend on pragmatic context. We benchmark various baselines and propose an end-to-end trainable neural model which takes into account multiple pragmatic aspects of the game state that may be commented upon to describe a given chess move. Through a human study on predictions for a subset of the data which deals with direct move descriptions, we observe that outputs from our models are rated similar to ground truth commentary texts in terms of correctness and fluency.[1]

1 Introduction

A variety of work in NLP has sought to produce fluent natural language descriptions conditioned on a contextual grounding. For example, several lines of work explore methods for describing images of scenes and videos (Karpathy and Fei-Fei, 2015), while others have conditioned on structured sources like Wikipedia infoboxes (Lebret et al., 2016). In most cases, progress has been driven by the availability of large training corpora that pair natural language with examples from the grounding (Lin et al., 2014). One line of work has investigated methods for producing and interpreting language in the context of a game, a space that has rich pragmatic structure, but where training data has been hard to come by. In this paper, we introduce a new large-scale resource for learning to correlate natural language with individual moves in the game of chess. We collect a dataset of more than 298K chess move/commentary pairs across ≈ 11K chess games from online chess forums. To the best of our knowledge, this is the first such dataset of this scale for a game commentary generation task. We provide an analysis of the dataset and highlight the large variety in commentary texts by categorizing them into six different aspects of the game that they respectively discuss.

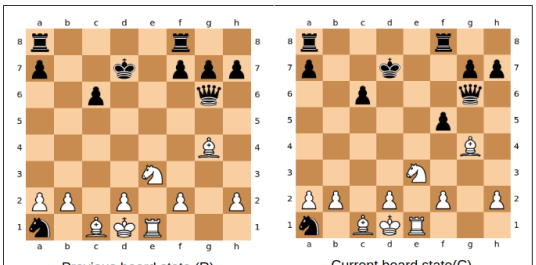

Figure 1: Move commentary generated from our method (Game-aware neural commentary generation (GAC)) and some baseline methods for a sample move.

Automated game commentary generation can be a useful learning aid. Novices and experts alike can learn more about the game by hearing expla-

[*] HJ and VG contributed equally for this paper

[1] We will make the code-base (including data collection and processing) publicly available at https://github.com/harsh19/ChessCommentaryGeneration

nations of the motivations behind moves, or their quality. In fact, on sites for game aficionados, these commentaries are standard features, speaking to their interestingness and utility as complements to concrete descriptions of the game boards themselves.

Game commentary generation poses a number of interesting challenges for existing approaches to language generation. First, modeling human commentary is challenging because human commentators rely both on their prior knowledge of game rules as well as their knowledge of effective strategy when interpreting and referring to the game state. Secondly, there are multiple aspects of the game state that can be talked about for a given move — the commentator's choice depends on the pragmatic context of the game. For example, for the move shown in Figure 1, one can comment simply that the pawn was moved, or one may comment on how the check was blocked by that move. Both descriptions are true, but the latter is most salient given the player's goal. However, sometimes, none of the aspects may stand out as being most salient, and the most salient aspect may even change from commentator to commentator. Moreover, a human commentator may introduce variations in the aspects he or she chooses to talk about, in order to reduce monotony in the commentary. This makes the dataset a useful testbed not only for NLG but also for related work on modeling pragmatics in language (Liu et al., 2016).

Prior work has explored game commentary generation. Liao and Chang (1990); Sadikov et al. (2006) have explored chess commentary generation, but for lack of large-scale training data their methods have been mainly rule-based. Kameko et al. (2015) have explored commentary generation for the game of Shogi, proposing a two-step process where salient terms are generated from the game state and then composed in a language model. In contrast, given the larger amount of training data available to us, our proposed model uses an end-to-end trainable neural architecture to predict commentaries given the game state. Our model conditions on semantic and pragmatic information about the current state and explicitly learns to compose, conjoin, and select these features in a recurrent decoder module. We perform an experimental evaluation comparing against baselines and variants of our model that ablate various aspects of our proposed architecture. Outputs on the 'Move Description' subset of data from our final model were judged by humans to be as good as human written ground truth commentaries on measures of fluency and correctness.

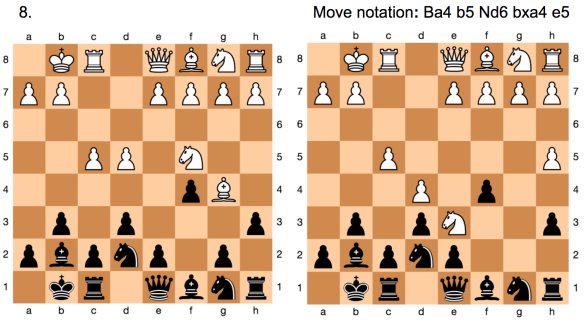

Figure 2: A multi-move, single commentary example from our data. Here, the sequence of moves Ba4 → b5 → Nd6 → bxa4 → e5 is commented upon.

Statistic	Value
Total Games	11,578
Total Moves	298,008
Average no. of recorded steps in a game	25.73
Frequent Word Types[2]	39,424
Rare Word Types	167,321
Word Tokens	6,125,921
Unigram Entropy	6.88
Average Comment Length (in #words)	20.55
Long Comments (#words)	230745 (77%)

Table 1: Dataset and Vocabulary Statistics

2 Chess Commentary Dataset

In this section we introduce our new large-scale *Chess Commentary* dataset, share some statistics about the data, and discuss the variety in type of commentaries. The data is collected from the online chess discussion forum `gameknot.com`, which features multiple games self-annotated with move-by-move commentary.

The dataset consists of 298K aligned game move/commentary pairs. Some commentaries are written for a sequence of few moves (Figure 2) while others correspond to a single move. For the purpose of initial analysis and modeling, we limit ourselves to only those data points where commentary text corresponds to a single move. Additionally, we split the multi-sentence commentary texts to create multiple data points with the same chess board and move inputs.

What are commentaries about? We observe that there is a large variety in the commentary

Category	Example	% in data	Val acc.
Direct Move Description	An attack on the queen	31.4%	71%
Move Quality	A rook blunder.	8.0%	90%
Comparative	At this stage I figured I better move my knight.	3.7%	77.7%
Planning / Rationale	Trying to force a way to eliminate d5 and prevent Bb5.	31.2%	65%
Contextual Game Info	Somehow, the game I should have lost turned around in my favor .	12.6%	87%
General Comment	Protect Calvin , Hobbs	29.9%	78%

Table 2: Commentary texts have a large variety making the problem of content selection an important challenge in our dataset. We classify the commentaries into 6 different categories using a classifier trained on some hand-labelled data, a fraction of which is kept for validation. % data refers to the percentage of commentary sentences in the tagged data belonging to the respective category.

texts. To analyze this variety, we consider labelling the commentary texts in the data with a predefined set of categories. The choice of these categories is made based on a manual inspection of a sub-sample of data. We consider the following set of commentary categories (Also shown in Table 2):

- **Direct move description (MoveDesc[3]):** Explicitly or implicitly describe the current move.
- **Quality of move (Quality[4]):** Describe the quality of the current move.
- **Comparative:** Compare multiple possible moves.
- **Move Rationale or Planning (Planning):** Describe the rationale for the current move, in terms of the future gameplay, advantage over other potential moves etc.
- **Contextual game information:** Describe not the current move alone, but the overall game state – such as possibility of win/loss, overall aggression/defence, etc.
- **General information:** General idioms & advice about chess, information about players/tournament, emotional remarks, retorts, etc.

The examples in Table 2 illustrate these classes. Note that the commentary texts are not necessarily limited to one tag, though that is true for most of the data. A total of 1K comments are annotated by two annotators. A SVM classifier (Pedregosa et al., 2011a) is trained for each comment class, considering the annotation as ground truth and using word unigrams as features. This classifier is then used to predict tags for the train, validation and test sets. For "Comparative" category, we found that a classifier with manually defined rules such as presence of word "better" performs better than the classifier, perhaps due to the paucity of data, and thus we use this instead . As can be observed in Table 2, the classifiers used are able to generalize well on the held out dataset

[3]MoveDesc & 'Move Description' used interchangeably
[4]Quality and 'Move Quality' used interchangeably

3 Game Aware Neural Commentary Generations (GAC)

Our dataset D consists of data points of the form $(S_i, M_i, G_i), i \in \{1, 2, .., |D|\}$, where S_i is the commentary text for move M_i and G_i is the corresponding chess game. S_i is a sequence of m tokens $S_{i1}, S_{i2}, ..., S_{im}$ We want to model $P(S_i|M_i, G_i)$. For simplicity, we use only current board (C_i) and previous board (R_i) information from the game. $P(S_i|M_i, G_i) = P(S_i|M_i, C_i, R_i)$.

We model this using an end-to-end trainable neural model, which models conjunctions of features using feature encoders. Our model employs a selection mechanism to select the salient features for a given chess move. Finally a LSTM recurrent neural network (Hochreiter and Schmidhuber, 1997) is used to generate the commentary text based on selected features from encoder.

3.1 Incorporating Domain Knowledge

Past work shows that acquiring domain knowledge is critical for NLG systems (Reiter et al., 2003b; Mahamood and Reiter, 2012). Commentary texts cover a range of perspectives, including criticism or goodness of current move, possible alternate moves, quality of alternate moves, etc. To be able to make such comments, the model must learn about the quality of moves, as well as the set of valid moves for a given chess board state. We consider the following features to provide our model with necessary information to generate commentary texts (Figure 3):

Move features $f_{move}(M_i, C_i, R_i)$ encode the current move information such as which piece moved, the position of the moved piece before and after the move was made, the type and position

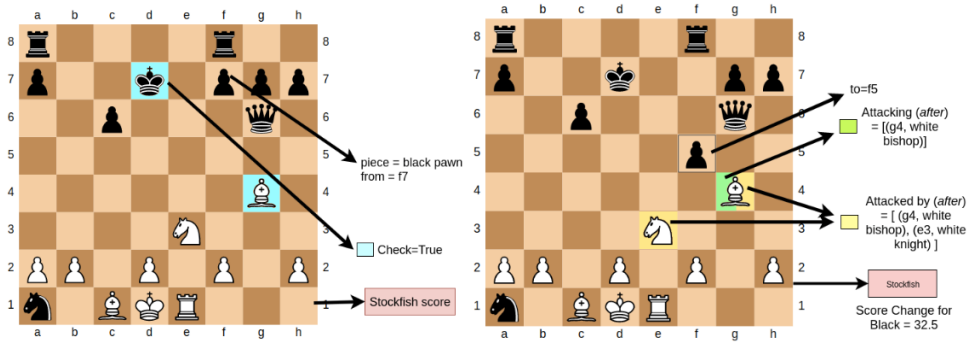

Figure 3: The figure shows some features extracted using the chess board states before (*left*) and after (*right*) a chess move. Our method uses various semantic and pragmatic features of the move, including the location and type of piece being moved, which opposing team pieces attack the piece being moved before as well as after the move, the change in score by *Stockfish* UCI engine, etc.

of the captured piece (if any), whether the current move is castling or not, and whether there was a check or not.

Threat features $f_{threat}(M_i, C_i, R_i)$ encode information about pieces of opposite player attacking the moved piece before and after the move, and the pieces of opposite player being attacked by the piece being moved. To extract this information, we use the *python-chess* library [5]

Score features $f_{score}(M_i, C_i, R_i)$ capture the quality of move and general progress of the game. This is done using the game evaluation score before and after the move, and average rank of pawns of both the players. We use *Stockfish* evaluation engine to obtain the game evaluation scores. [6]

3.2 Feature Representation

In our simplest conditioned language generation model **GAC-sparse**, we represent the above described features using sparse representations through binary-valued features. $g_{sparse}(M_i, C_i, R_i) = \text{SparseRep}(f_{move}, f_{threat}, f_{score})$

For our full **GAC** model we consider representing features through embeddings. This has the advantage of allowing for a shared embedding space, which is pertinent for our problem since attribute values can be shared, e.g. the same piece type can occur as the moved piece as well as the captured piece. For categorical features, such as those indicating which piece was moved, we directly look up the embedding using corresponding token. For real valued features such as game scores, we first bin them and then use corresponding number for embedding lookup. Let E represent the embedding matrix. Then $E[f^j_{move}]$ represents embeddings of j^{th} move feature, or in general $E[f_{move}]$ represents the concatenated embeddings of all move features. Similarly, $E(f_{move}, f_{threat}, f_{score})$ represents concatenated embeddings of all the features.

3.3 Feature Conjunctions

We conjecture that explicitly modeling feature conjunctions might improve the performance. So we need an encoder which can handle input sets of features of variable length (features such as pieces attacking the moved piece can be of variable length). One way to handle this is by picking up a canonical ordering of the features and consider a bidirectional LSTM encoder over the feature embeddings. As shown in Figure 4, this generates conjunctions of features.

$g^{enc} = \text{BiLSTM}^*(\{E(f_{move}, f_{threat}, f_{score}))\})$

Here $E()$ represents the embedding matrix as described earlier and $BiLSTM^*$ represents a sequential application of the $BiLSTM$ function. Thus, if there a total of m feature keys and embedding dimension is d, $E(f_{move}, f_{threat}, f_{score})$ is matrix of $m * d$. If hidden size of BILSTM is of size x, then g^{enc} is of dimensionality $m * x$. We observe that different orderings gave similar performance. We also experimented with running k encoders, each on different ordering of features, and then letting the decoder access to each of the k encodings. This did not yield any significant gain in performance.

The GAC model, unlike GAC-sparse, has some advantages as it uses a shared, continuous space

[5] https://pypi.org/project/python-chess/

[6] https://stockfishchess.org/about/

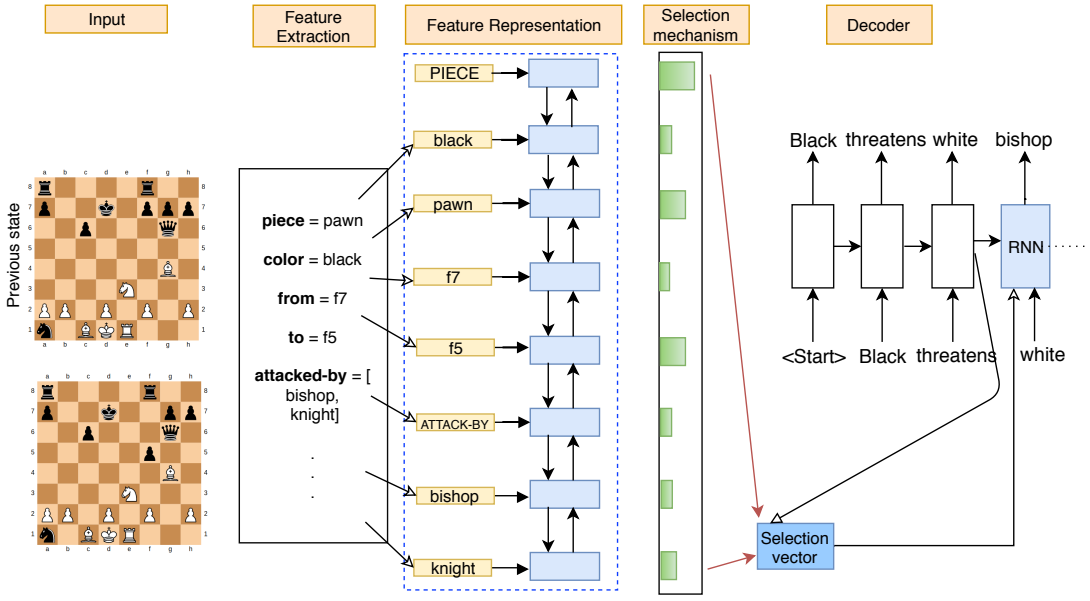

Figure 4: The figure shows a model overview. We first extract various semantic and pragmatic features from the previous and current chess board states. We represent features through embedding in a shared space. We observe that feeding in feature conjunctions helps a lot. We consider a selection mechanism for the model to choose salient attributes from the input at every decoder step.

to embed attribute values of different features, and can perform arbitrary feature conjunctions before passing a representation to the decoder, thereby sharing the burden of learning the necessary feature conjunctions. Our experiments confirm this intuition — GAC produces commentaries with higher BLEU as well as more diversity compared to GAC-sparse.

3.4 Decoder

We use a LSTM decoder to generate the sentence given the chess move and the features g. At every output step t, the LSTM decoder predicts a distribution over vocabulary words taking into account the current hidden state h_t, the input token i_t, and additional selection vector c_t. For GAC-sparse, the selection vector is simply an affine transformation of the features g. For GAC model selection vector is derived via a selection mechanism.

$$o_t, h_t^{dec} = LSTM(h_{t-1}^{dec}, [\text{concat}(E_{dec}(i_t), c_t)])$$
$$p_t = \text{softmax}(W_o[\text{concat}(o_t, c_t)] + b_s)$$

where p_t represents th probability distribution over the vocabulary, $E_{dec}()$ represents the decoder word embedding matrix and elements of W_o matrix are trainable parameters.

Selection/Attention Mechanism: As there are different salient attributes across the different chess moves, we also equip the GAC model with a mechanism to select and identify these attributes. We first transform h_t^{dec} by multiplying it with a trainable matrix W_c, and then take dot product of the result with each g_i.

$$a_t^{(i)} = dot(W_c * h_t^{dec}, g_i^{enc})$$
$$\alpha_t = \text{softmax}(a_t)$$
$$c_t = \sum_{i=1}^{i=|g|} \alpha_t^{(i)} g_i^{enc}$$

We use cross-entropy loss over the decoding outputs to train the model.

4 Experiments

We split each of the data subsets in a 70:10:20 ratio into train, validation and test. All our models are implemented in Pytorch version 0.3.1 (Paszke et al., 2017). We use the ADAM optimizer (Kingma and Ba, 2014) with its default parameters and a mini-batch size of 32. Validation set perplexity is used for early-stopping. At test-time, we use greedy search to generate the model output. We observed that beam decoding does not lead to any significant improvement in terms of validation BLEU score.

We observe the BLEU (Papineni et al., 2002) and BLEU-2 (Vedantam et al., 2015) scores to measure the performance of the models. Addi-

tionally, we consider a measure to quantify the diversity in the generated outputs. Finally, we also conduct a human evaluation study. In the remainder of this section, we discuss baselines along with various experiments and results.

4.1 Baselines

In this subsection we discuss the various baseline methods.

Manually-defined template (TEMP) We devise manually defined templates (Reiter, 1995) for 'Move Description' and 'Move Quality' categories. Note that template-based outputs tend to be repetitive as they lack diversity - drawing from a small, fixed vocabulary and using a largely static sentence structure. We define templates for a fixed set of cases which cover our data (For exact template specifications, refer to Appendix B).

Nearest Neighbor (NN): We observe that the same move on similar board states often leads to similar commentary texts. To construct a simple baseline, we find the most similar move N_{MCR} from among training data points for a given previous (R) and current (C) board states and move M. The commentary text corresponding to N_{MCR} is selected as the output. Thus, we need to consider a scoring function to find the closest matching data point in training set. We use the *Move*, *Threat* and *Score* features to compute similarity to do so. By using a sparse representation, we consider total of 148 *Move* features, 18 *Threat* features, and 19 *Score* features. We use sklearn's (Pedregosa et al., 2011b) NearestNeighbor module to find the closest matching game move.

Raw Board Information Only (RAW): The RAW baseline ablates to assess the importance of our pragmatic feature functions. This architecture is similar to GAC, except that instead of our custom features $A(f(R_i, C_i))$, the encoder encodes raw board information of current and previous board states.

$A_{RAW}(R_i, C_i) = [Lin(R_i), Lin(C_i)]$

$Lin()$ for a board denotes it's representation in a row-linear fashion. Each element of $Lin()$ is a piece name (e.g *pawn*) denoting the piece at that square with special symbols for empty squares.

4.2 Comment Category Models

As shown earlier, we categorize comments into six different categories. Among these, in this paper we consider only the first three as the amount of variance in the last three categories indicates that it would be extremely difficult for a model to learn to reproduce them accurately. The number of data points, as tagged by the trained classifiers, in the subsets 'Move Description', 'Move Quality' and 'Comparative' are 28,228, 793 and 5397 respectively. We consider separate commentary generation models for each of the three categories. Each model is tuned separately on the corresponding validation sets. Table 3 shows the BLEU and BLEU-2 scores for the proposed model under different subsets of features. Overall BLEU scores are low, likely due to the inherent variance in the language generation task (Novikova et al., 2017) , although a precursory examination of the outputs for data points selected randomly from test set indicated that they were reasonable. Figure 5 illustrates commentaries generated by our models through an example (a larger list of qualitative examples can be found in Appendix C).

Dataset	Features	BLEU	BLEU-2	Diversity
MoveDesc	TEMP	0.72	20.77	4.43
	NN (M+T+S)	1.28	21.07	**7.85**
	RAW	1.13	13.74	2.37
	GAC-sparse	1.76	21.49	4.29
	GAC (M+T)	**1.85**	**23.35**	**4.72**
Quality	TEMP	16.17	47.29	1.16
	NN (M+T)	5.98	42.97	4.52
	RAW	16.92	47.72	1.07
	GAC-sparse	14.98	51.46	2.63
	GAC(M+T+S)	16.94	47.65	1.01
Comparative	NN (M)	1.28	24.49	6.97
	RAW	2.80	23.26	3.03
	GAC-sparse	**3.58**	25.28	2.18
	GAC(M+T)	3.51	29.48	3.64

Table 3: Performance of baselines and our model with different subsets of features as per various quantitative measures. (**S** = Score, **M**= Move, **T** = Threat features;) On all data subsets, our model outperforms the **TEMP** and **NN** baselines. Among proposed models, GAC performs better than GAC-sparse & RAW in general. For NN, GAC-sparse and GAC methods, we experiment with multiple feature combinations and report only the best as per BLEU scores.

Which features are useful? In general, adding *Threat* features improves the performance, though the same is not always true for *Score* features. *Qual* has higher BLEU scores than the other datasets due to smaller vocabulary and lesser variation in commentary. As can be observed in Table 4, *Threat* features are useful for both 'Move Quality' and 'Move Description' subsets of data. Adding *Score* features helps for 'Move Quality' subset. This intuitively makes sense since *Score*

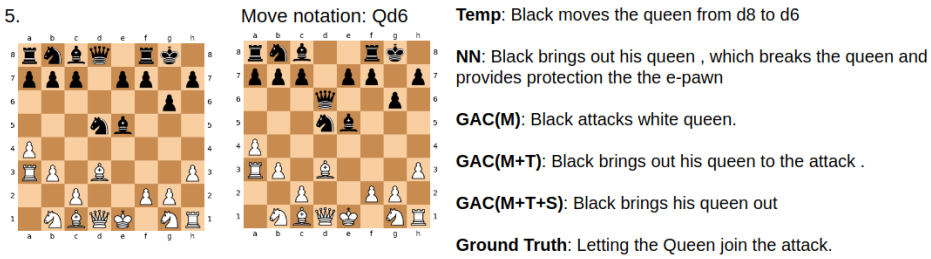

Figure 5: Outputs from various models on a test example from the **MoveDesc** subset.

Dataset	Features	BLEU	BLEU-2	Diversity
MoveDesc	GAC (M)	1.41	19.06	4.32
	GAC (M+T)	**1.85**	**23.35**	**4.72**
	GAC (M+T+S)	1.64	22.82	4.29
Quality	GAC (M)	13.05	48.37	1.61
	GAC (M+T)	14.22	49.57	**1.54**
	GAC(M+T+S)	**14.44**	**51.79**	1.48
Comparative	GAC(M)	3.10	19.84	2.88
	GAC(M+T)	**3.51**	**29.48**	**3.64**
	GAC(M+T+S)	1.15	25.44	3.14

Table 4: Performance of the GAC model with different feature sets. (**S** = Score, **M**= Move, **T** = Threat features;) Different subset of features work best for different subsets. For instance, *Score* features seem to help only in the Quality category. Note that the results for Quality are from 5-fold cross-validation, since the number of datapoints in the category is much lesser than the other two.

features directly encode proxies for move quality as per a chess evaluation engine.

4.3 A Single Model For All Categories

In this experiment, we merge the training and validation data of the first three categories and tune a single model for this merged data. We then compare its performance on all test sentences in our data. **COMB** denotes using the best GAC model for a test example based on its original class (e.g *Desc*) and computing the BLEU of the sentences so generated with the ground truth. **GAC-all** represents the GAC model learnt on the merged training data.

As can be seen from Table 5, this does not lead to any performance improvements. We investigate this issue further by analyzing whether the board states are predictive of the type of category or not. To achieve this, we construct a multi-class classifier using all the *Move*, *Threat* and *Score* features to predict the three categories under consideration. However, we observe accuracy of around 33.4%, which is very close to the performance of a random prediction model. This partially explains why a single model did not fare better even though it had the opportunity to learn

Dataset	Features	BLEU	BLEU-2	Diversity
All	COMB (M)	2.07	20.13	4.50
	COMB (M+T)	**2.43**	**25.37**	4.88
	COMB (M+T+S)	1.83	28.86	4.33
All	GAC-all(M)	1.69	20.66	4.67
	GAC-all(M+T)	1.94	24.11	5.16
	GAC-all (M+T+S)	2.02	24.70	4.97
All	CAT (M)	1.90	19.96	3.82

Table 5: The **COMB** approaches show the combined performance of separately trained models on the respective test subsets.

from a larger dataset.

Category-aware model (CAT) We observed above that with the considered features, it is not possible to predict the type of comment to be made, and the GAC-all model results are better than COMB results. Hence, we extend the GAC-all model to explicitly provide with the information about the comment category. We achieve this by adding a one-hot representation of the category of the comment to the input of the RNN decoder at every time step. As can be seen in the Table 5, CAT(M) performs better than GAC-all(M) in terms of BLEU-4, while performing slightly worse on BLEU-2. This demonstrates that explicitly providing information about the comment category can help the model.

4.4 Diversity In Generated Commentaries

Humans use some variety in the choice of words and sentence structure. As such, outputs from rule based templates, which demonstrate low variety, may seem repetitive and boring. To capture this quantitatively, and to demonstrate the variety in texts from our method, we calculate the entropy (Shannon, 1951) of the distribution of unigrams, bigrams and trigrams of words in the predicted outputs, and report the geometric mean of these values. Using only a small set of words in similar counts will lead to lower entropy and is undesirable. As can be observed from Table 3, template

baseline performs worse on the said measure compared to our methods for the 'MoveDesc' subset of the data.

4.5 Human Evaluation Study

As discussed in the qualitative examples above, we often found the outputs to be good - though BLEU scores are low. BLEU is known to correlate poorly (Reiter and Belz, 2009; Wiseman et al., 2017; Novikova et al., 2017) with human relevance scores for NLG tasks. Hence, we conduct a human evaluation study for the best 2 neural (GAC,GAC-sparse) and best 2 non-neural methods (TEMP,NN).

Setup: Specifically, annotators are shown a chess move through previous board and resulting board snapshots, along with information on which piece moved (a snapshot of a HIT[7] is provided in the Appendix D). With this context, they were shown text commentary based on this move and were asked to judge the commentary via three questions, shortened versions of which can be seen in the first column of Table 6.

We randomly select 100 data points from the test split of 'Move Description' category and collect the predictions from each of the methods under consideration. We hired two Anglophone (Lifetime HIT acceptance $\% > 80$) annotators for every human-evaluated test example. We additionally assess chess proficiency of the annotators using questions from the chess-QA dataset by (Cirik et al., 2015). Within each HIT, we ask two randomly selected questions from the chess-QA dataset. Finally we consider only those HITs wherein the annotator was able to answer the proficiency questions correctly.

Results: We conducted a human evaluation study for the *MoveDesc* subset of the data. As can be observed from Table 6, outputs from our method attain slightly more favorable scores compared to the ground truth commentaries. This shows that the predicted outputs from our model are not worse than ground truth on the said measures. This is in spite of the fact that the BLEU-4 score for the predicted outputs is only ~ 2 w.r.t. the ground truth outputs. One reason for slightly lower performance of the ground truth outputs on the said measures is that some of the human written commentaries are either very ungrammatical or too concise. A more surprising observation is that around 30% of human written ground truth outputs were also marked as not valid for given board move. On inspection, it seems that commentary often contains extraneous game information beyond that of move alone, which indicates that an ideal comparison should be over commentary for an entire game, although this is beyond the scope of the current work.

The inter-annotator agreement for our experiments (Cohens κ (Cohen, 1968)) is 0.45 for Q1 and 0.32 for Q2. We notice some variation in κ coefficients across different systems. While TEMP and GAC responses had a 0.5-0.7 coefficient range, the responses for CLM had a much lower coefficient. In our setup, each HIT consists of 7 comments, one from each system. For Q3 (fluency), which is on an ordinal scale, we measure rank-order consistency between the responses of the two annotators of a HIT. Mean Kendall τ (Kendall, 1938) across all HITs was found to be 0.39.

To measures significance of results, we perform bootstrap tests on 1000 subsets of size 50 with a significance threshold of $p = 0.05$ for each pair of systems. For Q1, we observe that GAC(M), GAC(M+T) and GAC(M+T+S) methods are significantly better than baselines NN and GAC-sparse. We find that neither of GAC(M+T) and GT significantly outperform each other on Q1 as well as Q2. But we do find that GAC(M+T) does better than GAC(M) on both Q1 and Q2. For fluency scores, we find that GAC(M+T) is more fluent than GT, NN , GAC-sparse, GAC(M). Neither of GAC(M) and GAC(M+T+S) is significantly more fluent than the other.

5 Related Work

NLG research has a long history, with systems ranging from completely rule-based to learning-based ones (Reiter et al., 2005, 2003a), which have had both practical successes (Reiter et al., 2005) and failures (Reiter et al., 2003a). Recently, there have been numerous works which propose text generation given structured records, biographies (Lebret et al., 2016), recipes (Yang et al., 2016; Kiddon et al., 2016), etc. A key difference between generation given a game state compared to these inputs is that the game state is an evolving description at a point in a process, as opposed

[7] Human Intelligence Task

Question	GT	GAC (M)	GAC (MT)	GAC (MTS)	GAC -sparse	TEMP	NN
Is commentary correct for the given move? (%Yes)	70.4	42.3	64.8	67.6	56.3	91.5	52.1
Can the move be inferred from the commentary? (%Yes)	45.1	25.3	42.3	36.7	40.8	92.9	42.3
Fluency (scale of (least)1 - 5(most)) Mean (Std. dev.)	4.03 (1.31)	4.15 (1.20)	4.44 (1.02)	4.54 (0.89)	4.15 (1.26)	4.69 (0.64)	3.72 (1.36)

Table 6: Human study results on *MoveDesc* data category. Outputs from GAC are in general better than ground truth, NN and GAC-sparse. TEMP outperforms other methods, though as shown earlier, outputs from TEMP lack diversity.

to recipes (which are independent of each other), records (which are static) and biographies (which are one per person, and again independent). Moreover, our proposed method effectively uses various types of semantic and pragmatic information about the game state.

In this paper we have introduced a new large-scale data for game commentary generation. The commentaries cover a variety of aspects like move description, quality of move, and alternative moves. This leads to a content selection challenge, similar to that noted in Wiseman et al. (2017). Unlike Wiseman et al. (2017), our focus is on generating commentary for individual moves in a game, as opposed to game summaries from aggregate statistics as in their task.

One of the first NLG datasets was the SUMTIME-METEO (Reiter et al., 2005) corpus with ≈ 500 record-text pairs for technical weather forecast generation. Liang et al (2009) worked on common weather forecast generation using the WEATHERGOV dataset, which has $\approx$ 10K record-text pairs. A criticism of WEATHERGOV dataset (Reiter, 2017) is that weather records themselves may have used templates and rules with optional human post-editing. There have been prior works on generating commentary for ROBOCUP matches (Chen and Mooney, 2008; Mei et al., 2015). The ROBOCUP dataset, however, is collected from 4 games and contains about 1K events in total. Our dataset is two orders of magnitude larger than the ROBOCUP dataset, and we hope that it provides a promising setting for future NLG research.

6 Conclusions

In this paper, we curate a dataset for the task of chess commentary generation and propose methods to perform generation on this dataset. Our proposed method effectively utilizes information related to the rules and pragmatics of the game. A human evaluation study judges outputs from the proposed methods to be as good as human written commentary texts for 'Move Description' subset of the data.

Our dataset also contains multi-move-single commentary pairs in addition to single move-single commentary pairs. Generating commentary for such multi-moves is a potential direction for future work. We anticipate this task to require even deeper understanding of the game pragmatics than the single move-single commentary case.

Recent work (Silver et al., 2016) has proposed reinforcement learning based game-playing agents which learn to play board games from scratch, learning end-to-end from both recorded games and self-play. An interesting point to explore is whether such pragmatically trained game state representations can be leveraged for the task of game commentary generation.

Acknowledgements

We thank Volkan Cirik, Daniel Clothiaux, Hiroaki Hayashi and anonymous reviewers for providing valuable comments and feedback.

References

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 128–135.

Volkan Cirik, Louis-Philippe Morency, and Eduard Hovy. 2015. Chess q&a: Question Answering on Chess Games. In *Reasoning, Attention, Memory (RAM) Workshop, Neural Information Processing Systems*.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin* 70(4):213.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Learning a game commentary generator with grounded move expressions. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*. IEEE, pages 177–184.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 3128–3137.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30(1/2):81–93.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 329–339.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771* .

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, pages 91–99.

Jen-Wen Liao and Jason S Chang. 1990. Computer Generation of Chinese Commentary on Othello Games. In *Proceedings of Rocling III Computational Linguistics Conference III*. pages 393–415.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, pages 740–755.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023* .

Saad Mahamood and Ehud Reiter. 2012. Working with clinicians to improve a patient-information NLG system. In *Proceedings of the Seventh International Natural Language Generation Conference*. Association for Computational Linguistics, pages 100–104.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838* .

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2241–2252. https://www.aclweb.org/anthology/D17-1238.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011a. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011b. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Ehud Reiter. 1995. NLG vs. templates. *arXiv preprint cmp-lg/9504013* .

Ehud Reiter. 2017. You Need to Understand Your Corpora - the Weathergov Example. *Blogpost - https://ehudreiter.com/2017/05/09/weathergov/* .

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics* 35(4):529–558.

Ehud Reiter, Roma Robertson, and Liesl M Osman. 2003a. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence* 144(1-2):41–58.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167(1-2):137–169.

Ehud Reiter, Somayajulu G Sripada, and Roma Robertson. 2003b. Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research* 18:491–516.

Aleksander Sadikov, Martin Moina, Matej Guid, Jana Krivec, and Ivan Bratko. 2006. Automated chess tutor. In *International Conference on Computers and Games*. Springer, pages 13–25.

Claude E Shannon. 1951. Prediction and entropy of printed English. *Bell Labs Technical Journal* 30(1):50–64.

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):484–489.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 4566–4575.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in Data-to-Document Generation. *arXiv preprint arXiv:1707.08052* .

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-Aware Language Models. *arXiv preprint arXiv:1611.01628* .