Generating Live Soccer-Match Commentary from Play Data

Yasufumi Taniguchi,[1] **Yukun Feng,**[1] **Hiroya Takamura,**[1,2] **Manabu Okumura**[1]
[1]Tokyo Institute of Technology
[2]National Institute of Advanced Industrial Science and Technology (AIST)
{yasufumi, yukun}@lr.pi.titech.ac.jp, {takamura, oku}@pi.titech.ac.jp

Abstract

We address the task of generating live soccer-match commentaries from play event data. This task has characteristics that (i) each commentary is only partially aligned with events, (ii) play event data contains many types of categorical and numerical attributes, (iii) live commentaries often mention player names and team names. For these reasons, we propose an encoder for play event data, which is enhanced with a gate mechanism. We also introduce an attention mechanism on events. In addition, we introduced placeholders and their reconstruction mechanism to enable the model to copy appropriate player names and team names from the input data. We conduct experiments on the play data of the English Premier League, provide a discussion on the result including generated commentaries.

Introduction

A soccer match consists of a series of numerous play events such as shots, passes, and fouls. Such events are coded as event descriptions (henceforth, *events*) as in Figure 1 and later used for broadcasting and match analysis. For example, `player_id=78412` in the figure means that the player mainly involved in this event is *Shinji Okazaki*, and `x=98.9 y=48.7` indicates the position where this event occurred. The first four lines in the figure provide the main information of the event and the remaining lines starting with `<Q` provide the additional detailed information. Coded events are used for broadcasting and match analysis.

Play event sequences are accompanied with live match commentaries. For example, a scene in a match Leicester vs. Newcastle in the 2015/16 season of the English premier league is described as follows:

> "*A scrappy goal from Okazaki who bundles the ball over the line after Elliot saves Simpson's effort.*"

This live commentary corresponds to a sequence of events, the last of which is the shot by Japanese striker *Okazaki* and is represented by the event data shown in Figure 1. We address the task of generating a live commentary from a sequence of events.

The task is regarded as a data-to-text generation task, but has characteristics that text is only partially aligned to events; it is not clear which events each commentary is aligned to. This characteristics is not specific to our experimental data. Live commentary data found in public is usually sparse, and not play-by-play data as seen in reddit[1] and the official webpage[2]. Other characteristics of this task are that input data consists of various types of information including categorical and numerical values, and that named entities (e.g., player names) are often mentioned in text. One motivation of this work is that it provides a solution to a data-to-text problem with similar characteristics. Another is that this work will be the first step towards generating more personalized live commentaries including those focusing on a particular player and those relating the viewpoint of the fans of one team. Although play events are currently coded by human workers, a lot of efforts are being made to automatize the work (e.g., (von Hoyningen-Huene 2011)), especially with the help of GPS and sensors attached to players (Liu et al. 2009; Buchheit et al. 2014).

The task of commentary generation contains many subtasks. In this paper, we address this task in a relaxed setting; we assume that which players to be mentioned and when to make a comment are given. We still need to work on content

```
<Event id="729781022" event_id="756" type_id="16" period_id="2"
min="82" sec="29" player_id="78412" team_id="13" outcome="1"
x="98.9" y="48.7" timestamp="2015-11-21T16:40:55.732"
last_modified="2015-11-26T14:29:19" version="1448548159295">
<Q id="860252266" qualifier_id="102" value="48.6"/>
<Q id="448074494" qualifier_id="22"/>
<Q id="2025850896" qualifier_id="280" value="ATTEMPT_SAVED"/>
<Q id="1474554908" qualifier_id="56" value="Center"/>
<Q id="631491044" qualifier_id="136"/>
<Q id="608793252" qualifier_id="282" value="13"/>
<Q id="1648074294" qualifier_id="78"/>
<Q id="1533115916" qualifier_id="103" value="0.6"/>
<Q id="2026835309" qualifier_id="16"/>
<Q id="2009738174" qualifier_id="231" value="48.7"/>
<Q id="871319258" qualifier_id="214"/>
<Q id="455253476" qualifier_id="281" value="40725"/>
<Q id="348252489" qualifier_id="20"/>
<Q id="502773375" qualifier_id="230" value="99.8"/> </Event>
```

Figure 1: Example of event data. The first four lines provide the main information about the event; the remaining lines provide the additional detailed information.

[1]For example, see https://www.reddit.com/r/soccer/comments/8mbxr7/match_thread_real_madrid_vs_liverpool_champions/.

[2]For example, see https://www.premierleague.com/match/22713.

selection, as well as on sentence planning and realization. We will discuss this point later in the paper.

Related Work

There are two types of research on text generation for sports matches. One is the generation of a summary for an entire match. For example, van der Lee et al. (2017) addressed the generation of soccer-match summaries separately for the fans of home and visiting teams. Many other researchers have worked on summary generation for different types of sports matches including American football (Barzilay and Lapata 2005), Australian football (Lareau, Dras, and Dale 2011), basketball (Wiseman, Shieber, and Rush 2017), and soccer (Bouayad-Agha, Casamayor, and Wanner 2011). The other type of research is live commentary generation, which we address in this paper. Tanaka-Ishii et al. (1998) and Chen et al. (2008) worked on this task, but with data of simulation soccer matches, in which both the input data and the commentaries are much simpler than ours. The data used in their work contains only player names and play types with timestamps, while the data used in our work contains more detailed information such as players' positions and the ball speed. Other researchers worked on live commentary generation from a set of posts to microblogs (Kubo et al. 2013; Edouard et al. 2017), not from play data. Live commentary generation has also been explored in the domain of chess, where the complete data describing the state of the game is readily available (Kameko, Mori, and Tsuruoka 2015; Jhamtani et al. 2018).

There are many pieces of conventional work on data-to-text generation tasks, where template-based approaches are often used. We would like readers to refer to a survey paper (Gatt and Krahmer 2018) for details. Our work is different from such conventional work in that our method is a trainable neural-network based model. Recent work on data-to-text generation includes product review generation (Dong et al. 2017) and biography generation (Lebret, Grangier, and Auli 2016; Liu et al. 2018; Hachey, Radford, and Chisholm 2017; Sha et al. 2018). Although our task is similar to these two kinds of tasks to a certain extent, it has its own characteristics that the input data is not well aligned with output text as discussed later. Another type of data-to-text generation task is text generation from a series of numerical values such as stock prices (Murakami et al. 2017) as opposed to the generation from tables as the two pieces of work mentioned above.

Wiseman et al. (2017) examined a number of datasets for data-to-text tasks including the summary generation for basketball matches. The task addressed in their paper is similar to, but different from ours in that their task is to generate a summary written from the statistics of the match after it ended, while our task is to generate live commentaries.

Play data of soccer

We use play event data of soccer matches in the English Premier League[3] for the 2015/16 season containing 380 soccer matches, provided by OptaSports.[4] The play data of each match consists of a sequence of events. An example of an event is shown in Figure 1. Each event consists of many pieces of information including play category, player names, ball position, time, height of ball, etc.

Table 1: Statistics of the dataset. The number of commentaries mentioning each number of player names. 5+ means 5 or more.

# of player names	1	2	3	4	5+
# of commentaries	6825	7613	2167	450	85

Table 2 shows some pieces of information described in the event in Figure 1 and their value types. Note that this is only a part of an event description, which actually contains a lot of more detailed information. There are 70 play categories designated by `type_id` (e.g., *pass*, *foul*, *attempt saved*, *clearance*), and 298 subcategories designated by `qualifier_id` (e.g., *long ball*, *through ball*, *lob*, *volley*). Although not all the information in this dataset can be automatically obtained with the current technology, efforts are being made to enable automatic recognition of play category and other information using image or video processing, and GPS technology (Liu et al. 2009; Buchheit et al. 2014) as argued in Introduction.

The original dataset provided by OptaSports contains 663,911 events and 26,340 commentaries. It means that there are a lot more events than commentaries. Also, on average, each match contains approximately 70 commentaries. Therefore, the commentaries in this dataset are not in the play-by-play style. Most of the events are ignored and only important events are described as commentaries. Table 1 provides the statistics of the dataset showing how many commentaries mention only one player name, two player names, and so on. The table shows that more than 60% of the commentaries contain multiple player names. It also shows that most commentaries mention three or less player names.

In this work, we address this generation task under a relaxed setting; we assume that which players to be mentioned and when to make a comment are given. We therefore conduct the following preprocessing on the data. For each live commentary, we first selected the events that contain the player names mentioned in the commentary and are time-stamped within 5 minutes before and after[5] the posting time of the commentary. From the selected events, we further selected the closest five events on the timeline and associate them with the commentary. We regarded such a pair of multiple events and a commentary as one *instance*. Even after this relaxation, the commentaries in the data are only partially aligned with events. In addition, each event contains many pieces of information, most of which are not mentioned in the commentaries. Therefore, although the content

[3]https://www.premierleague.com

[4]https://www.optasports.com The example commentary in Introduction and the example in Figure 1 were also provided by OptaSports.

[5]The reason we also use events *after* the commentary is that the time associated with each commentary is sometimes deviated from the time associated with each event.

Table 2: Example attributes describing the event in Figure 1. Note that this is only a part of an event description, which actually contains a lot of more detailed information.

attribute	example attribute value	value type
player name	*Shinji Okazaki*	categorical
play category	goal	categorical
time	82min 29sec	continuous
x-y coordinates of the ball	98.9, 48.7	continuous
details	keeper touched, big chance, fantasy assisted	categorical

selection is partially done through the relaxation, the task addressed in this work still contains content selection, as well as on sentence planning and realization.

On the other hand, the commentaries sometimes describe information beyond the input play data. For example, "*Wenger is furious with Noble on the touchline.*" is found in a commentary, although the input play data does not contain any information whether or not Arsène Wenger, the then head coach of Arsenal F.C., was furious. Some other commentaries contain expressions that are difficult (though not impossible) to generate such as "*scrappy goal*" and "*deadly cross*".

Live commentary generation

We use an encoder-decoder model, which receives an event sequence as input $(x_1, x_2, \ldots, x_n)$, and generates a live commentary $(y_1, y_2, \ldots, y_m)$ from the output of the encoder (Sutskever, Vinyals, and Le 2014), where x_i is an event and y_j is a word.

As an encoder, we use the multilayer perceptron (MLP), which performed best in Murakami et al. (2017). Since each input is a sequence of events, one might think that a recurrent neural network would work well as an encoder, i.e., a sequence-to-sequence model as a whole (Sutskever, Vinyals, and Le 2014). However, according to our observation, the actual inputs do not have the characteristics as sequences; they are rather sets of events. In fact, a sequence-to-sequence model did not work well in our preliminary experiments. We therefore focus on MLP in our work.

As a decoder, we use the recurrent neural network language model (RNNLM) (Bahdanau, Cho, and Bengio 2014) with long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997).

Figure 2 shows the neural network architecture of our model, which will be explained in detail in this section.

Encoding events

We use a mapping f to convert each event in input sequence $(x_1, x_2, \ldots, x_n)$ to a vector representation:

$$\boldsymbol{p}_i = f(x_i), \tag{1}$$

where x_i denotes an event consisting of a number of categorical and continuous values. Categorical values are represented as embeddings, and continuous values are preprocessed. Specifically, x and y coordinates of ball positions, which range from 0 to 100, are divided by 100 and normalized to [0,1]. Time at which the event occurs is converted to relative time; the delivery time of the commentary is subtracted from it. Additional detailed information (i.e., the lines with <Q in Figure 1) is first represented as a bag-of-feature vector and then fed into the MLP, to obtain its vector representation. All the embedding vectors of categorical values, the preprocessed continuous values, and the vector representations of additional detailed information are concatenated to make a vector $\boldsymbol{p}_i$.

The sequence $P = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$ is fed into the encoder, in which $\boldsymbol{p}_i$ are again concatenated and applied to the MLP to obtain the internal representation $\boldsymbol{h}$:

$$\boldsymbol{h} = MLP([\boldsymbol{p}_1; \boldsymbol{p}_2; \ldots; \boldsymbol{p}_n]), \tag{2}$$

where [] represents a vector concatenation. We use the batch normalization (Ioffe and Szegedy 2015) in the MLP of the encoder. The internal representation $\boldsymbol{h}$ is then fed into the decoder, which generates a live commentary. Yang et al. (2016) reported that the model's ability improves when the encoder is trained to predict whether each word appears in the output sentence or not, in addition to the sequence prediction. We borrow their idea and specifically focus on the content words; in addition to the sequence prediction, we train our model to predict whether each content word appears in the output sentence. For this purpose, we use the probability distribution over content words:

$$\boldsymbol{g}_c = \sigma(U_c \boldsymbol{h} + \boldsymbol{b}_c), \tag{3}$$

where U_c is a weight parameter matrix, $\boldsymbol{b}_c$ is a bias term vector, and $\sigma(\cdot)$ is a sigmoid function. Each element g_{ck} is the probability that word k appears in the output. As part of the total loss, we compute the sum of cross-entropies:

$$loss_1 = -\sum_{k \in C} \{t_{ck} \log g_{ck} + (1 - t_{ck}) \log(1 - g_{ck})\},$$

where t_{ck} is 1 if word k appears in the output and otherwise is 0.

Attention mechanism and placeholder reconstruction

The decoder generates word sequence $(y_1, y_2, \ldots, y_m)$ from the last hidden state $\boldsymbol{h}$ of the encoder using the probability over words:

$$p(y_j | y_{<j}, \boldsymbol{h}, P) = \mathrm{softmax}(W_s \tilde{\boldsymbol{s}}_j), \tag{4}$$

$$\tilde{\boldsymbol{s}}_j = \tanh(W_c[\boldsymbol{s}_j; \boldsymbol{c}_j]), \tag{5}$$

where $\boldsymbol{s}_j$ is the j-th hidden state of the decoder, and W_s and W_c are weight matrices. Vector $\boldsymbol{c}_j$ is the context vector

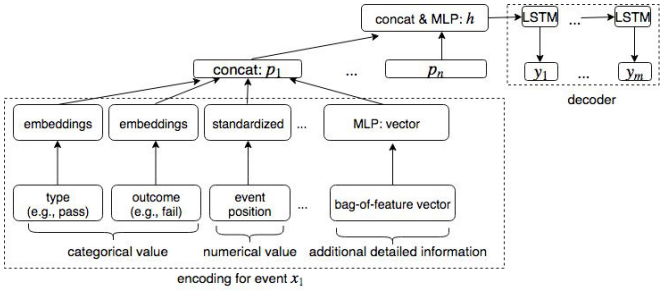

Figure 2: Neural network architecture of our model. Encoding of event x_1 as $\boldsymbol{p}_1$ is described in detail, but other events are similarly encoded. Vectors $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n$ are fed into MLP to calculate $\boldsymbol{h}$. For simplicity, attention mechanism is omitted in this figure.

of the j-th hidden state, and is used as part of the attention mechanism on input events $P = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$, defined as:

$$\boldsymbol{c}_j = \sum_{i=1}^{n} a_{ij} \boldsymbol{p}_i, \quad (6)$$

$$a_{ij} = \frac{\exp(score(\boldsymbol{p}_i, \boldsymbol{s}_j))}{\sum_{k=1}^{n} \exp(score(\boldsymbol{p}_k, \boldsymbol{s}_j))}, \quad (7)$$

where a_{ij} is the alignment probability between the i-th event and the j-th hidden state in the decoder. This attention mechanism is introduced, because the model needs to rely on different events adaptively when it generates each word. As $score$, we use the MLP[6] following the work by Luong et al. (2015).

Live commentaries usually contain many mentions of named entities such as players and teams. Since it is difficult for the model to learn all the named entities, we replace such mentions with *placeholders* such as `player` and `team`, which retain the semantic classes of the mentions. Similar ideas can be found in the literature (Lebret, Grangier, and Auli 2016; Murakami et al. 2017). In addition, we append a number to each placeholder to distinguish player names or team names that appear in the same commentary. For example, the commentary

> "*A scrappy goal from Okazaki who bundles the ball over the line after Elliot saves Simpson's effort.*"

is converted to

> "*A scrappy goal from* `player_1` *who bundles the ball over the line after* `player_2` *saves* `player_3` *'s effort.*"

The decoder generates a sequence possibly containing placeholders, which need to be converted back to the terms. This is not a trivial problem because there can be multiple candidates. To solve this problem, we use the attention score following the idea of the copy mechanism. For each `player`, for example, we choose the event with the highest attention score a_{ij}, and replace the placeholder with the name of the player mainly involved in this event.

[6]In Luong et al. (2015), this function is referred to as *concat*. We refrained from using the term *concat* to avoid the confusion with the word "concatenation" later in the paper.

We further attempt to force the attention score to point to the correct named entity that should be replaced with a placeholder. In particular, we add the following term to the loss function:

$$loss_2 = \sum_{j=1}^{m} \sum_{k \in U_j} a_{kj}. \quad (8)$$

If j-th token y_j in the output is a placeholder, U_j is defined to be the set of indices to events that do not contain the player or team name associated with placeholder y_j. If y_j is not a placeholder, U_j is defined to be the empty set. This loss function will make the model attend to the events containing the entities to be replaced with y_j. Finally, we minimize the sum of $loss_1$, $loss_2$, and the negative log-likelihood of the training commentaries, which is the standard loss function.

To distinguish players involved in different types of plays, we also test fine-grained placeholders. If a live commentary in the training data contains expressions (e.g., *dribble*) in the right column in Table 3, we replace the player name in the commentary with the corresponding fine-grained placeholder (e.g., `player-dribble`). When a commentary has multiple mentions of player names of the same type, we number them as is done above; we end up with, for example, `player-dribble_1` and `player-dribble_2`.

Table 3: Fine-grained placeholders and associated key expressions

placeholder	expressions in text
`player-save-from`	*save from*
`player-fire`	*fire*
`player-cross`	*cross*
`player-free-kick`	*free-kick*
`player-forced`	*forced*
`player-caught`	*caught*
`player-release`	*release*
`player-shot`	*shot*
`player-chance`	*chance*
`player-dribble`	*dribble*
`player-shoot`	*shoot*
`player-go-close`	*go close*
`player-other`	other than above

Gate mechanism for the encoder

Although the attention mechanism controls which event to look at when generating each word, we also incorporate another mechanism to distinguish important events without referring to the hidden states in the decoder. In particular, we introduce a gate mechanism (Zhou et al. 2017), in which the internal representation $\boldsymbol{h}$ computed from the input events by Equation (2) in the section of Encoding events. is used as the aggregated information of the play data sequence, to calculate the importance score γ_i for event i:

$$\gamma_i = \sigma(\boldsymbol{w}_g \cdot [\boldsymbol{h}; \boldsymbol{p}_i]), \quad (9)$$

where $\boldsymbol{w}_g$ is a weight vector. γ_i is a scalar used to make a new event vector $\boldsymbol{p}'_i = \gamma_i \boldsymbol{p}_i$. Therefore, the information of $\boldsymbol{p}_i$ is regarded as important when the value of γ_i is close to 1, while the information is mostly ignored when γ_i is close to 0. After this gate mechanism is applied, we apply the attention mechanism mentioned earlier to the event sequence $P' = (\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_n)$

Word concatenation

The overfitting problem is more serious when the training data is small, as in our case where we only have 17,140 pairs of multiple events and a live commentary. The trained model tends to excessively generate expressions that are frequent in the training data, such as "*is replaced*" and "*is booked*". Once the model generates "*is*", it tends to predict "*replaced*" or "*booked*" next regardless of the contents of the input. To address this problem, we concatenate frequent bigrams and regard them as a single word. Specifically, we concatenate the most-frequent 100 bigrams in the training set, including "*is replaced*" and "*is booked*".

Experiments

Experimental settings

We used 17,140 commentaries as our experimental data, out of which 13,662 were used as training data, 1,677 as development data, and 1,801 as test data. We used nltk[7] for tokenizing the live commentary data, and the neural-network framework Chainer[8] to implement our model.[9] For the parameter optimization, we used Adam (Kingma and Ba 2014) and set the gradient clipping value (Pascanu, Mikolov, and Bengio 2012) to 5. We stopped the model training if the BLEU score on the development data became worse three times in a row, or the number of epochs reached 200. We set the dimension of the embedding vectors for the categorical values in events to 16, that of encoder's hidden state $\boldsymbol{h}$ to 200, and that of the decoder's word embedding vectors to 128. Each parameter is initialized using the Xavier initialization method (Glorot and Bengio 2010). During training, we truncate each commentary to 10 words.

[7] http://www.nltk.org

[8] https://chainer.org

[9] Although the data itself has to be purchased, the code for preprocessing data and other related resources are available at https://github.com/yasufumy/placeholder_reconstruction.

We conducted both automatic evaluation and human evaluation. For automatic evaluation, we used BLEU (Papineni et al. 2002) between the generated live commentaries and the gold-standard text. Since the reference commentary is not the only commentary that is correct especially in data-to-text generation tasks, BLEU scores based on reference commentary might not be perfectly accurate. However, we conjecture that BLEU scores can provide rough estimations on the trend of model performance. We compensate what is missing in BLEU evaluation with human evaluation.

We compared the models described in Table 4 to see whether each component (i.e., gate mechanism, placeholder, reconstruction error, and word concatenation) contributes to the result or not. Note that we show only the models that we think are useful for examining the trend. The models are numbered 1 through 7 and 7'. Model 1, which does not contain any of those components, is regarded as the baseline in this paper. Model 7 has all the components above. Only Model 7' uses fine-grained placeholders (e.g., `player-dribble`), while Models 2, 3, 4, 5, and 7 use coarse-grained placeholders, i.e., `player` and `team`.

We also note that adding the reconstruction error without placeholder does not make sense, because the reconstruction error is added to correctly replace the placeholders with the original player names and team names.

For human evaluation, we asked workers at Amazon Mechanical Turk[10] to give a score between 1 and 3 for both the *grammaticality* and the *informativeness* of the output for 100 instances. For each instance, 10 workers were assigned to conduct this evaluation task. In the evaluation of informativeness, the workers were supposed to compare the generated commentaries with the gold-standard ones to measure the informativeness. The reason is that reading play data would be significantly difficult for non-expert workers; this is a general problem in the evaluation in the data-to-text generation task when the input data is very complicated. We also note that the gold-standard commentaries are not evaluated because it does not make sense to compare the gold-standard commentaries with themselves.

Results and Discussions

We show the experimental results in Table 4. In addition to the BLEU scores on the full test set, the table also shows the BLEU scores for the sentences that were shorter than or equal to 20, 15, and 10. The following is observed from the BLEU scores and human evaluation.

- Models 2 to 7 outperformed Model 1, i.e., the baseline.
- Comparison between Models 1 and 2 suggests that the use of placeholder improves the performance.
- Comparison between Models 2 and 4, also between Models 3 and 5 suggests that adding the reconstruction error term to the loss function further improves the performance. The degradation of the performance of Model 6 compared with Model 7 also supports the benefit of using reconstruction error.

[10] https://www.mturk.com

Table 4: Evaluation result (BLEU scores and human evaluation). Models are numbered 1 through 7 and 7'. Model 1 is the baseline method. The components (gate mechanism, placeholder, reconstruction error, word concatenation) of each model is shown in the row 'components'. Only Model 7' uses fine-grained placeholders. BLEU scores are calculated for test instances with different lengths; e.g., $l \leq$20 means that the lengths (the number of words) of the commentaries are equal to or less than 20. The best BLEU scores are written in bold font. Human evaluation scores are the averages of the scores given by human evaluators. The scores with † are statistically significantly better than that of the baseline (Model 1) in t-test with significance level 0.05.

models		1	2	3	4	5	6	7	7'
components	gate mechanism			✓		✓	✓	✓	✓
	placeholder		✓	✓	✓	✓		✓	✓
	reconstruction error				✓	✓		✓	✓
	word concatenation						✓	✓	✓
BLEU scores	$l \leq \infty$	0.11	0.43	0.58	0.65	0.51	0.12	**0.69**	0.57
	$l \leq$20	0.23	1.30	1.52	1.83	1.83	0.34	1.90	**1.93**
	$l \leq$15	0.28	2.77	2.93	3.58	3.87	0.65	4.07	**4.13**
	$l \leq$10	0.45	4.78	5.19	6.38	**7.46**	0.86	6.79	6.76
human evaluation	grammaticality	1.65	1.71	1.89†	1.88†	1.99†	1.34	2.07†	2.13†
	informativeness	1.41	1.38	1.38	1.58†	1.60†	1.16	1.69†	1.67†

- Comparison between Models 2 and 3, and also between Models 4 and 5 suggests that the gate mechanism also contributes to the improvement in performance.
- The difference between Models 7 and 7' is not significant in terms of BLEU and human evaluation. It suggests that fine-grained placeholders do not provide extra benefit as opposed to coarse-grained placeholders.

We conclude that the components introduced in this paper contribute to the improvement of generation performance, although fine-grained placeholders did not provide extra benefit to coarse-grained placeholders. However, there is much room for further improvement.

In Table 5, we show some examples of sentences generated by Models 1 (baseline) and 7' (with all components and fine-grained placeholders), in addition to the gold-standard reference commentaries. The following is observed from these examples.

- In the first example with *Harry Kane*, while the sentence generated by Model 1 does not make sense, the sentence generated by Model 7' successfully describes *Harry Kane*'s attempt to score a goal although it does not mention the penalty area.
- In the second example with *Odion Ighalo*, the live commentary generated by Model 7' correctly describes the play events although its expression is different from that of the reference live commentary.
- In Examples 3 and 4 in Table 5, Model 7' correctly generated two different player names, e.g., *Chris Smalling* and *Dwight Gayle*. However, Model 7' fails to correctly distinguish player names as in Example 5, in which *Cameron Borthwick-Jackson* should be the player who sends a ball.
- Also in Example 7, Model 7' mentions the same player name twice resulting in a commentary that does not make sense. Errors of this type are caused by the drawback of our placeholder mechanism. Our model retrieves the name of the player who is the main agent in the most-attended event. If all the main agents are a single player, our model has no choice but to select this player, even though selecting this player twice makes the commentary uninterpretable. This is one limitation of our model and should be modified in future work.
- In Example 8, although *corner kick* is mentioned in the reference commentary, Models 1 and 7' mention *free kick*. This happens because events are not well aligned with the commentary and the models used a wrong event to generate a commentary.
- In general, Model 1 tends to generate ungrammatical sentences as in Example 3 ("*Blind is played in behind by an exquisite throw of play by the.*").

These observations lead to important issues to be solved in future work, which will be summarized in Conclusion.

Conclusion

We proposed a method for generating live soccer-match commentaries from play event data. To generate player names and team names correctly, we introduced placeholders and their reconstruction mechanism. We also introduced a gate mechanism to select important events. We obtained favorable results in experiments.

Although we succeeded in aligning events with commentaries by introducing reconstruction error to a certain extent, the alignment is still far from perfect and there are many errors caused by such imperfect alignments as we discussed in the section of Results and Discussions. More powerful technique for this problem would improve the generation performance even further.

In the present work, we assumed that play event data is given. However, some pieces of information in the event data might be hard to obtain automatically. We will examine the play data carefully to distinguish automatically obtainable attributes from the others. We will then work on the

Table 5: Reference live commentaries written by human (ref.) and generated live commentaries of Model 1 (baseline) and Model 7' (with all the components and fine-grained placeholders)

	date	teams	time	model	live commentary
1	Feb 6, 2016	Tottenham vs. Watford	52nd	ref.	*Harry Kane goes down inside the penalty area.*
				1	*Half-hearted Everton lead breaks forward through an early side with.*
				7'	*Harry Kane is trying to score for Tottenham Hotspur.*
2	Mar 5, 2016	Watford vs. Leicester	80th	ref.	*Huge chance missed by Odion Ighalo.*
				1	*Watford are finally made a winner now as they are bringing.*
				7'	*Odion Ighalo has a shot blocked.*
3	Oct 31, 2015	Crystal Palace vs. Manchester U.	47th	ref.	*Chris Smalling is shown yellow for a foul on Dwight Gayle.*
				1	*Blind is played in behind by an exquisite throw of play by the.*
				7'	*Chris Smalling is shown a yellow card for a foul on Dwight Gayle.*
4	Aug 8, 2015	Chelsea vs. Swansea City	41st	ref.	*Sung-Yueng Ki indeed goes off and he is replaced by Jack Cork.*
				1	*A change for Bournemouth as Ki comes on for Ki.*
				7'	*Sung-Yueng Ki is replaced by Jack Cork for Swansea City.*
5	Jan 17, 2016	Liverpool vs. Manchester U.	56th	ref.	*Cameron Borthwick-Jackson sends an inviting ball into the area from the left but Wayne Rooney flubs at it.*
				1	*A cross by Rooney results in a dangerous ball that.*
				7'	*Cameron Borthwick-Jackson is given a long ball into the path of Wayne Rooney.*
6	Mar 5, 2016	Everton vs. West Ham U.	90th	ref.	*What a turn-around from the visits as Dimitri Payet gets on the end of a knock-down to slot a finish under Joel Robles.*
				1	*A second yellow card of the Arsenal penalty area from the right and clips.*
				7'	*Dimitri Payet gets a free kick in from the left-hand side of the West Ham United.*
7	Oct 19, 2015	Swansea City vs. Stoke City	57th	ref.	*Jonjo Shelvey committed a late foul on Joselu in the build up and he is booked.*
				1	*Swansea have had a vital cynical with Shelvey and put.*
				7'	*Jonjo Shelvey is shown a yellow card for a foul on Jonjo Shelvey.*
8	Mar 2, 2016	Arsenal vs. Swansea City	58th	ref.	*Ozil whips a corner into the area, and Alexis takes it onto his chest before prodding the ball for the bottom corner but Williams deflects it wide.*
				1	*A late free kick here for Skrtel now as the two man.*
				7'	*Ashley Williams is given a free kick into the game for Swansea City.*

live commentary generation task under a more realistic task setting.

We also assumed that players to be mentioned are given. In some situations where we would like to generate live commentaries focusing on a certain player, this assumption is realistic. However, when we generate general live commentaries, the assumption is not realistic and we also need to determine on which player a live commentary should be generated, and also when in a match. Incorporating movies and images of the matches is also a part of future work.

Acknowledgments This work was supported by JST PRESTO (Grant Number JPMJPR1655).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv*:1409.0473.

Barzilay, R., and Lapata, M. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 331–338.

Bouayad-Agha, N.; Casamayor, G.; and Wanner, L. 2011. Content selection from an ontology-based knowledge base for the generation of football summaries. In *Proceedings of the 13th European Workshop on Natural Language Generation*, 72–81.

Buchheit, M.; Allen, A.; Poon, T. K.; Modonutti, M.; Gregson, W.; and Di Salvo, V. 2014. Integrating different tracking systems in football: multiple camera semi-automatic system, local position measurement and GPS technologies. *Journal of Sports Sciences* 32(20):1844–1857.

Chen, D. L., and Mooney, R. J. 2008. Learning to sportscast: a test of grounded language acquisition. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, 128–135.

Dong, L.; Huang, S.; Wei, F.; Lapata, M.; Zhou, M.; and Xu, K. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 1, 623–632.

Edouard, A.; Cabrio, E.; Le Thanh, N.; and Tonelli, S. 2017. You'll Never Tweet Alone Building Sports Match Timelines from Microblog Posts. In *Proceedings of Recent Advances in Natural Language Processing 17*, 214–221.

Gatt, A., and Krahmer, E. 2018. Survey of the state of the art in natural language generation: Core tasks, applications

and evaluation. *Journal of Artificial Intelligence Research* 61(1):65–170.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 249–256.

Hachey, B.; Radford, W.; and Chisholm, A. 2017. Learning to generate one-sentence biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 633–642.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 448–456.

Jhamtani, H.; Gangal, V.; Hovy, E.; Neubig, G.; and Berg-Kirkpatrick, T. 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)*, 1661–1671.

Kameko, H.; Mori, S.; and Tsuruoka, Y. 2015. Learning a game commentary generator with grounded move expressions. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 177–184.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv*:1412.6980.

Kubo, M.; Sasano, R.; Takamura, H.; and Okumura, M. 2013. Generating live sports updates from twitter by finding good reporters. In *Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, 527–534.

Lareau, F.; Dras, M.; and Dale, R. 2011. Detecting interesting event sequences for sports reporting. In *Proceedings of the 13th European Workshop on Natural Language Generation*, 200–205.

Lebret, R.; Grangier, D.; and Auli, M. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1203–1213.

Liu, J.; Tong, X.; Li, W.; Wang, T.; Zhang, Y.; and Wang, H. 2009. Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern Recognition Letters* 30(2):103–113.

Liu, T.; Wang, K.; Sha, L.; Chang, B.; and Sui, Z. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 4881–4888.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1412–1421.

Murakami, S.; Watanabe, A.; Miyazawa, A.; Goshima, K.; Yanase, T.; Takamura, H.; and Miyao, Y. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 1374–1384.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 311–318.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2012. Understanding the exploding gradient problem. *arXiv*:1211.5063.

Sha, L.; Mou, L.; Liu, T.; Poupart, P.; Li, S.; Chang, B.; and Sui, Z. 2018. Order-planning neural text generation from structured data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 5414–5421.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *arXiv*:1409.3215.

Tanaka-Ishii, K.; Hasida, K.; and Noda, I. 1998. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL-COLING)*, 1282–1288.

van der Lee, C.; Krahmer, E.; and Wubben, S. 2017. Pass: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, 95–104.

von Hoyningen-Huene, N. 2011. *Real-time Tracking of Player Identities in Team Sports*. Ph.D. Dissertation, Fakultät für Informatik der Technischen Universität München.

Wiseman, S.; Shieber, S. M.; and Rush, A. M. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2253–2263.

Yang, Z.; Yuan, Y.; Wu, Y.; Cohen, W. W.; and Salakhutdinov, R. 2016. Review networks for caption generation. In *Advances in Neural Information Processing Systems 29*, 2361–2369.

Zhou, Q.; Yang, N.; Wei, F.; and Zhou, M. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 1095–1104.