Learning to Select, Track, and Generate for Data-to-Text

Hayate Iso[†*] **Yui Uehara**[‡] **Tatsuya Ishigaki**[♮‡] **Hiroshi Noji**[‡]
Eiji Aramaki[†‡] **Ichiro Kobayashi**[♭‡] **Yusuke Miyao**[♯‡] **Naoaki Okazaki**[♮‡] **Hiroya Takamura**[♮‡]
[†]Nara Institute of Science and Technology [‡]Artificial Intelligence Research Center, AIST
[♮]Tokyo Institute of Technology [♭]Ochanomizu University [♯]The University of Tokyo
{iso.hayate.id3,aramaki}@is.naist.jp koba@is.ocha.ac.jp
{yui.uehara,ishigaki.t,hiroshi.noji,takamura.hiroya}@aist.go.jp
yusuke@is.s.u-tokyo.ac.jp okazaki@c.titech.ac.jp

Abstract

We propose a data-to-text generation model with two modules, one for tracking and the other for text generation. Our tracking module selects and keeps track of salient information and memorizes which record has been mentioned. Our generation module generates a summary conditioned on the state of tracking module. Our model is considered to simulate the human-like writing process that gradually selects the information by determining the intermediate variables while writing the summary. In addition, we also explore the effectiveness of the writer information for generation. Experimental results show that our model outperforms existing models in all evaluation metrics even without writer information. Incorporating writer information further improves the performance, contributing to content planning and surface realization.

1 Introduction

Advances in sensor and data storage technologies have rapidly increased the amount of data produced in various fields such as weather, finance, and sports. In order to address the information overload caused by the massive data, data-to-text generation technology, which expresses the contents of data in natural language, becomes more important (Barzilay and Lapata, 2005). Recently, neural methods can generate high-quality short summaries especially from small pieces of data (Liu et al., 2018).

Despite this success, it remains challenging to generate a high-quality long summary from data (Wiseman et al., 2017). One reason for the difficulty is because the input data is too large for a naive model to find its salient part, i.e., to determine which part of the data should be mentioned. In addition, the salient part moves as the summary explains the data. For example, when generating a summary of a basketball game (Table 1 (b)) from the box score (Table 1 (a)), the input contains numerous data records about the game: e.g., *Jordan Clarkson scored 18 points*. Existing models often refer to the same data record multiple times (Puduppully et al., 2019). The models may mention an incorrect data record, e.g., *Kawhi Leonard added 19 points*: the summary should mention *LaMarcus Aldridge*, who scored 19 points. Thus, we need a model that finds salient parts, tracks transitions of salient parts, and expresses information faithful to the input.

In this paper, we propose a novel data-to-text generation model with two modules, one for saliency tracking and another for text generation. The tracking module keeps track of saliency in the input data: when the module detects a saliency transition, the tracking module selects a new data record[1] and updates the state of the tracking module. The text generation module generates a document conditioned on the current tracking state. Our model is considered to imitate the human-like writing process that gradually selects and tracks the data while generating the summary. In addition, we note some writer-specific patterns and characteristics: how data records are selected to be mentioned; and how data records are expressed as text, e.g., the order of data records and the word usages. We also incorporate writer information into our model.

The experimental results demonstrate that, even without writer information, our model achieves the best performance among the previous models in all evaluation metrics: 94.38% precision of relation generation, 42.40% F1 score of content selection, 19.38% normalized Damerau-Levenshtein

[*]Work was done during the internship at Artificial Intelligence Research Center, AIST

[1]We use 'data record' and 'relation' interchangeably.

Distance (DLD) of content ordering, and 16.15% of BLEU score. We also confirm that adding writer information further improves the performance.

2 Related Work

2.1 Data-to-Text Generation

Data-to-text generation is a task for generating descriptions from structured or non-structured data including sports commentary (Tanaka-Ishii et al., 1998; Chen and Mooney, 2008; Taniguchi et al., 2019), weather forecast (Liang et al., 2009; Mei et al., 2016), biographical text from infobox in Wikipedia (Lebret et al., 2016; Sha et al., 2018; Liu et al., 2018) and market comments from stock prices (Murakami et al., 2017; Aoki et al., 2018).

Neural generation methods have become the mainstream approach for data-to-text generation. The encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014) with the attention (Bahdanau et al., 2015; Luong et al., 2015) and copy mechanism (Gu et al., 2016; Gulcehre et al., 2016) has successfully applied to data-to-text tasks. However, neural generation methods sometimes yield fluent but inadequate descriptions (Tu et al., 2017). In data-to-text generation, descriptions inconsistent to the input data are problematic.

Recently, Wiseman et al. (2017) introduced the RotoWire dataset, which contains multi-sentence summaries of basketball games with box-score (Table 1). This dataset requires the selection of a salient subset of data records for generating descriptions. They also proposed automatic evaluation metrics for measuring the informativeness of generated summaries.

Puduppully et al. (2019) proposed a two-stage method that first predicts the sequence of data records to be mentioned and then generates a summary conditioned on the predicted sequences. Their idea is similar to ours in that the both consider a sequence of data records as content planning. However, our proposal differs from theirs in that ours uses a recurrent neural network for saliency tracking, and that our decoder dynamically chooses a data record to be mentioned without fixing a sequence of data records.

2.2 Memory modules

The memory network can be used to maintain and update representations of the salient information (Weston et al., 2015; Sukhbaatar et al., 2015; Graves et al., 2016). This module is often used in natural language understanding to keep track of the entity state (Kobayashi et al., 2016; Hoang et al., 2018; Bosselut et al., 2018).

Recently, entity tracking has been popular for generating coherent text (Kiddon et al., 2016; Ji et al., 2017; Yang et al., 2017; Clark et al., 2018). Kiddon et al. (2016) proposed a neural checklist model that updates predefined item states. Ji et al. (2017) proposed an entity representation for the language model. Updating entity tracking states when the entity is introduced, their method selects the salient entity state.

Our model extends this entity tracking module for data-to-text generation tasks. The entity tracking module selects the salient entity and appropriate attribute in each timestep, updates their states, and generates coherent summaries from the selected data record.

3 Data

Through careful examination, we found that in the original dataset RotoWire, some NBA games have two documents, one of which is sometimes in the training data and the other is in the test or validation data. Such documents are similar to each other, though not identical. To make this dataset more reliable as an experimental dataset, we created a new version.

We ran the script provided by Wiseman et al. (2017), which is for crawling the RotoWire website for NBA game summaries. The script collected approximately 78% of the documents in the original dataset; the remaining documents disappeared. We also collected the box-scores associated with the collected documents. We observed that some of the box-scores were modified compared with the original RotoWire dataset.

The collected dataset contains 3,752 instances (i.e., pairs of a document and box-scores). However, the four shortest documents were not summaries; they were, for example, an announcement about the postponement of a match. We thus deleted these 4 instances and were left with 3,748 instances. We followed the dataset split by Wiseman et al. (2017) to split our dataset into training, development, and test data. We found 14 instances that didn't have corresponding instances in the original data. We randomly classified 9, 2, and 3 of those 14 instances respectively into training, development, and test data. Finally, the sizes of

TEAM	H/V	WIN	LOSS	PTS	REB	AST	FG_PCT	FG3_PCT	...
KNICKS	H	16	19	104	46	26	45	46	...
BUCKS	V	18	16	105	42	20	47	32	...

PLAYER	H/V	PTS	REB	AST	BLK	STL	MIN	CITY	...
CARMELO ANTHONY	H	30	11	7	0	2	37	NEW YORK	...
DERRICK ROSE	H	15	3	4	0	1	33	NEW YORK	...
COURTNEY LEE	H	11	2	3	1	1	38	NEW YORK	...
GIANNIS ANTETOKOUNMPO	V	27	13	4	3	1	39	MILWAUKEE	...
GREG MONROE	V	18	9	4	1	3	31	MILWAUKEE	...
JABARI PARKER	V	15	4	3	0	1	37	MILWAUKEE	...
MALCOLM BROGDON	V	12	6	8	0	0	38	MILWAUKEE	...
MIRZA TELETOVIC	V	13	1	0	0	0	21	MILWAUKEE	...
JOHN HENSON	V	2	2	0	0	0	14	MILWAUKEE	...
...	...	...	...	...	...	...	...		

(a) Box score: Top contingency table shows number of wins and losses and summary of each game. Bottom table shows statistics of each player such as points scored (PLAYER's PTS), and total rebounds (PLAYER's REB).

The **Milwaukee Bucks** defeated the **New York Knicks**, 105-104, at Madison Square Garden on Wednesday. The **Knicks** (16-19) checked in to Wednesday's contest looking to snap a five-game losing streak and heading into the fourth quarter, they looked like they were well on their way to that goal. . . . **Antetokounmpo** led the Bucks with 27 points, 13 rebounds, four assists, a steal and three blocks, his second consecutive double-double. **Greg Monroe** actually checked in as the second-leading scorer and did so in his customary bench role, posting 18 points, along with nine boards, four assists, three steals and a block. **Jabari Parker** contributed 15 points, four rebounds, three assists and a steal. **Malcolm Brogdon** went for 12 points, eight assists and six rebounds. **Mirza Teletovic** was productive in a reserve role as well, generating 13 points and a rebound. . . . **Courtney Lee** checked in with 11 points, three assists, two rebounds, a steal and a block. . . . The Bucks and Knicks face off once again in the second game of the home-and-home series, with the meeting taking place Friday night in Milwaukee.

(b) NBA basketball game summary: Each summary consists of game victory or defeat of the game and highlights of valuable players.

Table 1: Example of input and output data: task defines box score (1a) used for input and summary document of game (1b) used as output. Extracted entities are shown in **bold face**. Extracted values are shown in green.

t	199	200	201	202	203	204	205	206	207	208	209
Y_t	Jabari	Parker	contributed	15	points	,	four	rebounds	,	three	assists
Z_t	1	1	0	1	0	0	1	0	0	1	0
E_t	JABARI PARKER	JABARI PARKER	-	JABARI PARKER	-	-	JABARI PARKER	-	-	JABARI PARKER	-
A_t	FIRST NAME	LAST NAME	-	PLAYER PTS	-	-	PLAYER REB	-	-	PLAYER AST	-
N_t	-	-	-	0	-	-	1	-	-	1	-

Table 2: Running example of our model's generation process. At every time step t, model predicts each random variable. Model firstly determines whether to refer to data records ($Z_t = 1$) or not ($Z_t = 0$). If random variable $Z_t = 1$, model selects entity E_t, its attribute A_t and binary variables N_t if needed. For example, at $t = 202$, model predicts random variable $Z_{202} = 1$ and then selects the entity **JABARI PARKER** and its attribute **PLAYER PTS**. Given these values, model outputs token **15** from selected data record.

our training, development, test dataset are respectively 2,714, 534, and 500. On average, each summary has 384 tokens and 644 data records. Each match has only one summary in our dataset, as far as we checked. We also collected the writer of each document. Our dataset contains 32 different writers. The most prolific writer in our dataset wrote 607 documents. There are also writers who wrote less than ten documents. On average, each writer wrote 117 documents. We call our new dataset ROTOWIRE-MODIFIED.[2]

4 Saliency-Aware Text Generation

At the core of our model is a neural language model with a memory state $\boldsymbol{h}^{\text{LM}}$ to generate a summary $y_{1:T} = (y_1, \ldots, y_T)$ given a set of data records $\boldsymbol{x}$. Our model has another memory state $\boldsymbol{h}^{\text{ENT}}$, which is used to remember the data records that have been referred to. $\boldsymbol{h}^{\text{ENT}}$ is also used to update $\boldsymbol{h}^{\text{LM}}$, meaning that the referred data records affect the text generation.

Our model decides whether to refer to $\boldsymbol{x}$, which data record $r \in \boldsymbol{x}$ to be mentioned, and how to express a number. The selected data record is used to update $\boldsymbol{h}^{\text{ENT}}$. Formally, we use the four variables:

1. Z_t: binary variable that determines whether the model refers to input $\boldsymbol{x}$ at time step t ($Z_t = 1$).
2. E_t: At each time step t, this variable indicates the salient entity (e.g., HAWKS, LEBRON JAMES).
3. A_t: At each time step t, this variable indicates the salient attribute to be mentioned (e.g., PTS).
4. N_t: If attribute A_t of the salient entity E_t is a numeric attribute, this variable determines if a value in the data records should be output in Arabic numerals (e.g., 50) or in English words (e.g., five).

To keep track of the salient entity, our model predicts these random variables at each time step

[2]For information about the dataset, please follow this link: https://github.com/aistairc/rotowire-modified

t through its summary generation process. Running example of our model is shown in Table 2 and full algorithm is described in Appendix A. In the following subsections, we explain how to initialize the model, predict these random variables, and generate a summary. Due to space limitations, bias vectors are omitted.

Before explaining our method, we describe our notation. Let $\mathcal{E}$ and $\mathcal{A}$ denote the sets of entities and attributes, respectively. Each record $r \in \boldsymbol{x}$ consists of entity $e \in \mathcal{E}$, attribute $a \in \mathcal{A}$, and its value $\boldsymbol{x}[e,a]$, and is therefore represented as $r = (e, a, \boldsymbol{x}[e,a])$. For example, the box-score in Table 1 has a record r such that $e =$ ANTHONY DAVIS, $a =$ PTS, and $\boldsymbol{x}[e,a] = 20$.

4.1 Initialization

Let $\boldsymbol{r}$ denote the embedding of data record $r \in \boldsymbol{x}$. Let $\bar{\boldsymbol{e}}$ denote the embedding of entity e. Note that $\bar{\boldsymbol{e}}$ depends on the set of data records, i.e., it depends on the game. We also use $\boldsymbol{e}$ for static embedding of entity e, which, on the other hand, does not depend on the game.

Given the embedding of entity $\boldsymbol{e}$, attribute $\boldsymbol{a}$, and its value $\boldsymbol{v}$, we use the concatenation layer to combine the information from these vectors to produce the embedding of each data record (e, a, v), denoted as $\boldsymbol{r}_{e,a,v}$ as follows:

$$\boldsymbol{r}_{e,a,v} = \tanh\left(\boldsymbol{W}^{\mathrm{R}}(\boldsymbol{e} \oplus \boldsymbol{a} \oplus \boldsymbol{v})\right), \tag{1}$$

where $\oplus$ indicates the concatenation of vectors, and $\boldsymbol{W}^{\mathrm{R}}$ denotes a weight matrix.[3]

We obtain $\bar{\boldsymbol{e}}$ in the set of data records $\boldsymbol{x}$, by summing all the data-record embeddings transformed by a matrix:

$$\bar{\boldsymbol{e}} = \tanh\left(\sum_{a \in \mathcal{A}} \boldsymbol{W}_a^{\mathrm{A}} \boldsymbol{r}_{e,a,\boldsymbol{x}[e,a]}\right), \tag{2}$$

where $\boldsymbol{W}_a^{\mathrm{A}}$ is a weight matrix for attribute a. Since $\bar{\boldsymbol{e}}$ depends on the game as above, $\bar{\boldsymbol{e}}$ is supposed to represent how entity e played in the game.

To initialize the hidden state of each module, we use embeddings of <SOD> for $\boldsymbol{h}^{\mathrm{LM}}$ and averaged embeddings of $\bar{\boldsymbol{e}}$ for $\boldsymbol{h}^{\mathrm{ENT}}$.

4.2 Saliency transition

Generally, the saliency of text changes during text generation. In our work, we suppose that the saliency is represented as the entity and its attribute being talked about. We therefore propose a model that refers to a data record at each time-point, and transitions to another as text goes.

To determine whether to transition to another data record or not at time t, the model calculates the following probability:

$$p(Z_t = 1 \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}}) = \sigma(\boldsymbol{W}_z(\boldsymbol{h}_{t-1}^{\mathrm{LM}} \oplus \boldsymbol{h}_{t-1}^{\mathrm{ENT}})), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function. If $p(Z_t = 1 \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}})$ is high, the model transitions to another data record.

When the model decides to transition to another, the model then determines which entity and attribute to refer to, and generates the next word (Section 4.3). On the other hand, if the model decides not transition to another, the model generates the next word without updating the tracking states $\boldsymbol{h}_t^{\mathrm{ENT}} = \boldsymbol{h}_{t-1}^{\mathrm{ENT}}$ (Section 4.4).

4.3 Selection and tracking

When the model refers to a new data record ($Z_t = 1$), it selects an entity and its attribute. It also tracks the saliency by putting the information about the selected entity and attribute into the memory vector $\boldsymbol{h}^{\mathrm{ENT}}$. The model begins to select the subject entity and update the memory states if the subject entity will change.

Specifically, the model first calculates the probability of selecting an entity:

$$p(E_t = e \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}}) \propto \begin{cases} \exp\left(\boldsymbol{h}_s^{\mathrm{ENT}} \boldsymbol{W}^{\mathrm{OLD}} \boldsymbol{h}_{t-1}^{\mathrm{LM}}\right) & \text{if } e \in \mathcal{E}_{t-1} \\ \exp\left(\bar{\boldsymbol{e}} \boldsymbol{W}^{\mathrm{NEW}} \boldsymbol{h}_{t-1}^{\mathrm{LM}}\right) & \text{otherwise} \end{cases}, \tag{4}$$

where $\mathcal{E}_{t-1}$ is the set of entities that have already been referred to by time step t, and s is defined as $s = \max\{s : s \le t-1, e = e_s\}$, which indicates the time step when this entity was last mentioned.

The model selects the most probable entity as the next salient entity and updates the set of entities that appeared ($\mathcal{E}_t = \mathcal{E}_{t-1} \cup \{e_t\}$).

If the salient entity changes ($e_t \neq e_{t-1}$), the model updates the hidden state of the tracking model $\boldsymbol{h}^{\mathrm{ENT}}$ with a recurrent neural network with a gated recurrent unit (GRU; Chung et al., 2014):

$$\boldsymbol{h}_t^{\mathrm{ENT}\prime} = \begin{cases} \boldsymbol{h}_{t-1}^{\mathrm{ENT}} & \text{if } e_t = e_{t-1} \\ \mathrm{GRU}^{\mathrm{E}}(\bar{\boldsymbol{e}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}}) & \text{else if } e_t \notin \mathcal{E}_{t-1} \\ \mathrm{GRU}^{\mathrm{E}}(\boldsymbol{W}_s^{\mathrm{S}} \boldsymbol{h}_s^{\mathrm{ENT}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}}) & \text{otherwise.} \end{cases} \tag{5}$$

[3]We also concatenate the embedding vectors that represents whether the entity is in home or away team.

Note that if the selected entity at time step t, e_t, is identical to the previously selected entity e_{t-1}, the hidden state of the tracking model is not updated.

If the selected entity e_t is new ($e_t \notin \mathcal{E}_{t-1}$), the hidden state of the tracking model is updated with the embedding $\bar{\boldsymbol{e}}$ of entity e_t as input. In contrast, if entity e_t has already appeared in the past ($e_t \in \mathcal{E}_{t-1}$) but is not identical to the previous one ($e_t \neq e_{t-1}$), we use $\boldsymbol{h}_s^{\text{ENT}}$ (i.e., the memory state when this entity last appeared) to fully exploit the local history of this entity.

Given the updated hidden state of the tracking model $\boldsymbol{h}_t^{\text{ENT}}$, we next select the attribute of the salient entity by the following probability:

$$\begin{aligned} & p(A_t = a \mid e_t, \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_t^{\text{ENT}'}) \\ & \propto \exp\left(\boldsymbol{r}_{e_t,a,\boldsymbol{x}[e_t,a]} \boldsymbol{W}^{\text{ATTR}} (\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}'})\right). \end{aligned} \quad (6)$$

After selecting a_t, i.e., the most probable attribute of the salient entity, the tracking model updates the memory state $\boldsymbol{h}_t^{\text{ENT}}$ with the embedding of the data record $\boldsymbol{r}_{e_t,a_t,\boldsymbol{x}[e_t,a_t]}$ introduced in Section 4.1:

$$\boldsymbol{h}_t^{\text{ENT}} = \text{GRU}^{\text{A}}(\boldsymbol{r}_{e_t,a_t,\boldsymbol{x}[e_t,a_t]}, \boldsymbol{h}_t^{\text{ENT}'}). \quad (7)$$

4.4 Summary generation

Given two hidden states, one for language model $\boldsymbol{h}_{t-1}^{\text{LM}}$ and the other for tracking model $\boldsymbol{h}_t^{\text{ENT}}$, the model generates the next word y_t. We also incorporate a copy mechanism that copies the value of the salient data record $\boldsymbol{x}[e_t, a_t]$.

If the model refers to a new data record ($Z_t = 1$), it directly copies the value of the data record $\boldsymbol{x}[e_t, a_t]$. However, the values of numerical attributes can be expressed in at least two different manners: Arabic numerals (e.g., *14*) and English words (e.g.,*fourteen*). We decide which one to use by the following probability:

$$p(N_t = 1 \mid \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_t^{\text{ENT}}) = \sigma(\boldsymbol{W}^{\text{N}}(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}})), \quad (8)$$

where $\boldsymbol{W}^{\text{N}}$ is a weight matrix. The model then updates the hidden states of the language model:

$$\boldsymbol{h}_t' = \tanh\left(\boldsymbol{W}^{\text{H}}(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}})\right), \quad (9)$$

where $\boldsymbol{W}^{\text{H}}$ is a weight matrix.

If the salient data record is the same as the previous one ($Z_t = 0$), it predicts the next word y_t via a probability over words conditioned on the context vector $\boldsymbol{h}_t'$:

$$p(Y_t \mid \boldsymbol{h}_t') = \text{softmax}(\boldsymbol{W}^{\text{Y}} \boldsymbol{h}_t'). \quad (10)$$

Subsequently, the hidden state of language model $\boldsymbol{h}^{\text{LM}}$ is updated:

$$\boldsymbol{h}_t^{\text{LM}} = \text{LSTM}(\boldsymbol{y}_t \oplus \boldsymbol{h}_t', \boldsymbol{h}_{t-1}^{\text{LM}}), \quad (11)$$

where $\boldsymbol{y}_t$ is the embedding of the word generated at time step t.[4]

4.5 Incorporating writer information

We also incorporate the information about the writer of the summaries into our model. Specifically, instead of using Equation (9), we concatenate the embedding $\boldsymbol{w}$ of a writer to $\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}}$ to construct context vector $\boldsymbol{h}_t'$:

$$\boldsymbol{h}_t' = \tanh\left(\boldsymbol{W}'^{\text{H}}(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}} \oplus \boldsymbol{w})\right), \quad (12)$$

where $\boldsymbol{W}'^{\text{H}}$ is a new weight matrix. Since this new context vector $\boldsymbol{h}_t'$ is used for calculating the probability over words in Equation (10), the writer information will directly affect word generation, which is regarded as surface realization in terms of traditional text generation. Simultaneously, context vector $\boldsymbol{h}_t'$ enhanced with the writer information is used to obtain $\boldsymbol{h}_t^{\text{LM}}$, which is the hidden state of the language model and is further used to select the salient entity and attribute, as mentioned in Sections 4.2 and 4.3. Therefore, in our model, the writer information affects both surface realization and content planning.

4.6 Learning objective

We apply fully supervised training that maximizes the following log-likelihood:

$$\begin{aligned} & \log p(Y_{1:T}, Z_{1:T}, E_{1:T}, A_{1:T}, N_{1:T} \mid \boldsymbol{x}) \\ & = \sum_{t=1}^{T} \log p(Z_t = z_t \mid \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_{t-1}^{\text{ENT}}) \\ & + \sum_{t:Z_t=1} \log p(E_t = e_t \mid \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_{t-1}^{\text{ENT}}) \\ & + \sum_{t:Z_t=1} \log p(A_t = a_t \mid e_t, \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_t^{\text{ENT}'}) \\ & + \sum_{t:Z_t=1, a_t \text{is num\_attr}} \log p(N_t = n_t \mid \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_t^{\text{ENT}}) \\ & + \sum_{t:Z_t=0} \log p(Y_t = y_t \mid \boldsymbol{h}_t') \end{aligned}$$

[4]In our initial experiment, we observed a word repetition problem when the tracking model is not updated during generating each sentence. To avoid this problem, we also update the tracking model with special trainable vectors $\boldsymbol{v}_{\text{REFRESH}}$ to refresh these states after our model generates a period: $\boldsymbol{h}_t^{\text{ENT}} = \text{GRU}^A(\boldsymbol{v}_{\text{REFRESH}}, \boldsymbol{h}_t^{\text{ENT}})$

Method	RG		CS			CO	BLEU
	#	P%	P%	R%	F1%	DLD%	
GOLD	27.36	93.42	100.	100.	100.	100.	100.
TEMPLATES	54.63	100.	31.01	58.85	40.61	17.50	8.43
Wiseman et al. (2017)	22.93	60.14	24.24	31.20	27.29	14.70	14.73
Puduppully et al. (2019)	33.06	83.17	33.06	43.59	37.60	16.97	13.96
PROPOSED	39.05	**94.43**	**35.77**	**52.05**	**42.40**	**19.38**	**16.15**

Table 3: Experimental result. Each metric evaluates whether important information (CS) is described accurately (RG) and in correct order (CO).

5 Experiments

5.1 Experimental settings

We used ROTOWIRE-MODIFIED as the dataset for our experiments, which we explained in Section 3. The training, development, and test data respectively contained 2,714, 534, and 500 games.

Since we take a supervised training approach, we need the annotations of the random variables (i.e., Z_t, E_t, A_t, and N_t) in the training data, as shown in Table 2. Instead of simple lexical matching with $r \in \boldsymbol{x}$, which is prone to errors in the annotation, we use the information extraction system provided by Wiseman et al. (2017). Although this system is trained on noisy rule-based annotations, we conjecture that it is more robust to errors because it is trained to minimize the marginalized loss function for ambiguous relations. All training details are described in Appendix B.

5.2 Models to be compared

We compare our model[5] against two baseline models. One is the model used by Wiseman et al. (2017), which generates a summary with an attention-based encoder-decoder model. The other baseline model is the one proposed by Puduppully et al. (2019), which first predicts the sequence of data records and then generates a summary conditioned on the predicted sequences. Wiseman et al. (2017)'s model refers to all data records every timestep, while Puduppully et al. (2019)'s model refers to a subset of all data records, which is predicted in the first stage. Unlike these models, our model uses one memory vector $\boldsymbol{h}_t^{\mathrm{ENT}}$ that tracks the history of the data records, during generation. We retrained the baselines on our new dataset. We also present the performance of the GOLD and TEMPLATES summaries. The GOLD summary is exactly identical with the reference summary and each TEMPLATES summary is generated in the same manner as Wiseman et al. (2017).

In the latter half of our experiments, we examine the effect of adding information about writers. In addition to our model enhanced with writer information, we also add writer information to the model by Puduppully et al. (2019). Their method consists of two stages corresponding to content planning and surface realization. Therefore, by incorporating writer information to each of the two stages, we can clearly see which part of the model to which the writer information contributes to. For Puduppully et al. (2019) model, we attach the writer information in the following three ways:

1. concatenating writer embedding $\boldsymbol{w}$ with the input vector for LSTM in the content planning decoder (stage 1);
2. concatenating writer embedding $\boldsymbol{w}$ with the input vector for LSTM in the text generator (stage 2);
3. using both 1 and 2 above.

For more details about each decoding stage, readers can refer to Puduppully et al. (2019).

5.3 Evaluation metrics

As evaluation metrics, we use BLEU score (Papineni et al., 2002) and the extractive metrics proposed by Wiseman et al. (2017), i.e., relation generation (RG), content selection (CS), and content ordering (CO) as evaluation metrics. The extractive metrics measure how well the relations extracted from the generated summary match the correct relations[6]:

[5]Our code is available from `https://github.com/aistairc/sports-reporter`

[6]The model for extracting relation tuples was trained on tuples made from the entity (e.g., team name, city name, player name) and attribute value (e.g., "Lakers", "92") ex-

- RG: the ratio of the correct relations out of all the extracted relations, where correct relations are relations found in the input data records $\boldsymbol{x}$. The average number of extracted relations is also reported.
- CS: precision and recall of the relations extracted from the generated summary against those from the reference summary.
- CO: edit distance measured with normalized Damerau-Levenshtein Distance (DLD) between the sequences of relations extracted from the generated and reference summary.

6 Results and Discussions

We first focus on the quality of tracking model and entity representation in Sections 6.1 to 6.4, where we use the model without writer information. We examine the effect of writer information in Section 6.5.

6.1 Saliency tracking-based model

As shown in Table 3, our model outperforms all baselines across all evaluation metrics.[7] One of the noticeable results is that our model achieves slightly higher RG precision than the gold summary. Owing to the extractive evaluation nature, the generated summary of the precision of the relation generation could beat the gold summary performance. In fact, the template model achieves 100% precision of the relation generations.

The other is that only our model exceeds the template model regarding F1 score of the content selection and obtains the highest performance of content ordering. This imply that the tracking model encourages to select salient input records in the correct order.

6.2 Qualitative analysis of entity embedding

Our model has the entity embedding $\bar{\boldsymbol{e}}$, which depends on the box score for each game in addition to static entity embedding $\boldsymbol{e}$. Now we analyze the difference of these two types of embeddings.

We present a two-dimensional visualizations of both embeddings produced using PCA (Pearson, 1901). As shown in Figure 1, which is the visualization of static entity embedding $\boldsymbol{e}$, the top-ranked players are closely located.

We also present the visualizations of dynamic entity embeddings $\bar{\boldsymbol{e}}$ in Figure 2. Although we did not carry out feature engineering specific to the NBA (e.g., whether a player scored double digits or not)[8] for representing the dynamic entity embedding $\bar{\boldsymbol{e}}$, the embeddings of the players who performed well for each game have similar representations. In addition, the change in embeddings of the same player was observed depending on the box-scores for each game. For instance, *LeBron James* recorded a double-double in a game on April 22, 2016. For this game, his embedding is located close to the embedding of *Kevin Love*, who also scored a double-double. However, he did not participate in the game on December 26, 2016. His embedding for this game became closer to those of other players who also did not participate.

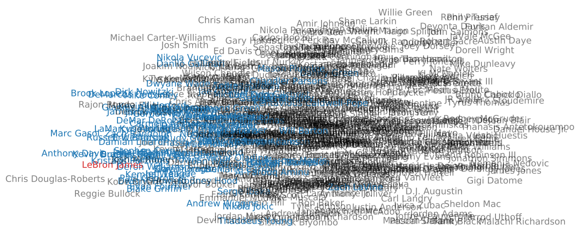

Figure 1: Illustrations of static entity embeddings $\boldsymbol{e}$. Players with colored letters are listed in the ranking top 100 players for the 2016-17 NBA season at https://www.washingtonpost.com/graphics/sports/nba-top-100-players-2016/. Only *LeBron James* is in red and the other players in top 100 are in blue. Top-ranked players have similar representations of $\boldsymbol{e}$.

6.3 Duplicate ratios of extracted relations

As Puduppully et al. (2019) pointed out, a generated summary may mention the same relation multiple times. Such duplicated relations are not favorable in terms of the brevity of text.

Figure 3 shows the ratios of the generated summaries with duplicate mentions of relations in the development data. While the models by Wiseman et al. (2017) and Puduppully et al. (2019) respec-

tracted from the summaries, and the corresponding attributes (e.g., "TEAM NAME", "PTS") found in the box- or line-score. The precision and the recall of this extraction model are respectively 93.4% and 75.0% in the test data.

[7]The scores of Puduppully et al. (2019)'s model significantly dropped from what they reported, especially on BLEU metric. We speculate this is mainly due to the reduced amount of our training data (Section 3). That is, their model might be more data-hungry than other models.

[8]In the NBA, a player who accumulates a *double*-digit score in one of five categories (points, rebounds, assists, steals, and blocked shots) in a game, is regarded as a good player. If a player had a double in two of those five categories, it is referred to as *double-double*.

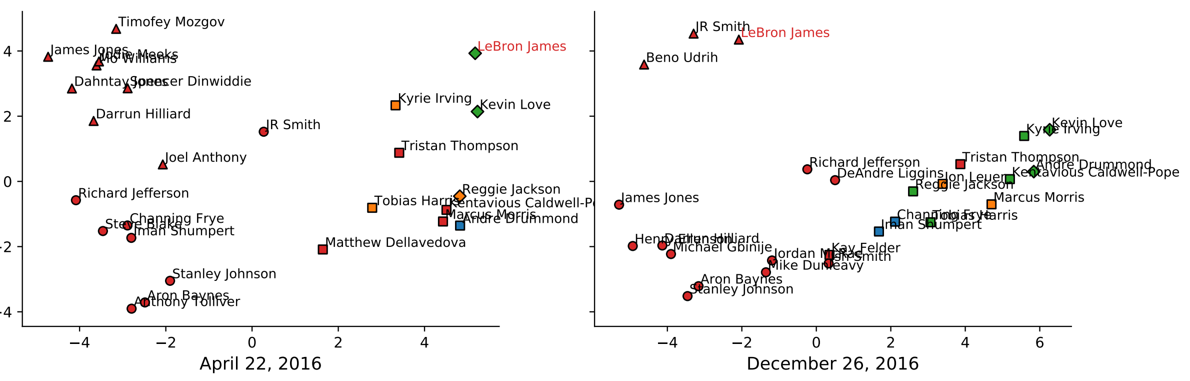

Figure 2: Illustrations of dynamic entity embedding $\bar{e}$. Both left and right figures are for *Cleveland Cavaliers* vs. *Detroit Pistons*, on different dates. LeBron James is in **red letters**. **Entities with orange symbols** appeared only in the reference summary. **Entities with blue symbols** appeared only in the generated summary. **Entities with green symbols** appeared in both the reference and the generated summary. The others are with **red symbols**. □ represents player who scored in the double digits, and ◇ represents player who recorded double-double. Players with △ did not participate in the game. ○ represents other players.

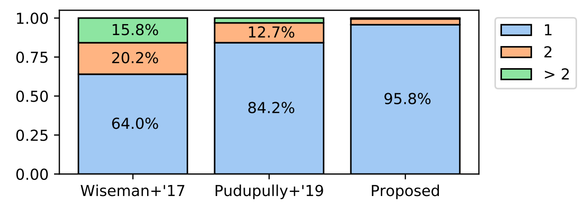

Figure 3: Ratios of generated summaries with duplicate mention of relations. Each label represents number of duplicated relations within each document. While Wiseman et al. (2017)'s model exhibited 36.0% duplication and Puduppully et al. (2019)'s model exhibited 15.8%, our model exhibited only 4.2%.

tively showed 36.0% and 15.8% as duplicate ratios, our model exhibited 4.2%. This suggests that our model dramatically suppressed generation of redundant relations. We speculate that the tracking model successfully memorized which input records have been selected in $\boldsymbol{h}_s^{\text{ENT}}$.

6.4 Qualitative analysis of output examples

Figure 5 shows the generated examples from validation inputs with Puduppully et al. (2019)'s model and our model. Whereas both generations seem to be fluent, the summary of Puduppully et al. (2019)'s model includes erroneous relations colored in orange.

Specifically, the description about DERRICK ROSE's relations, "15 points, four assists, three rounds and one steal in 33 minutes.", is also used for other entities (e.g., JOHN HENSON and WILLY HERNAGOMEZ). This is because Puduppully et al. (2019)'s model has no tracking module unlike our model, which mitigates redundant references and therefore rarely contains erroneous relations.

However, when complicated expressions such as parallel structures are used our model also generates erroneous relations as illustrated by the underlined sentences describing the two players who scored the same points. For example, "11-point efforts" is correct for COURTNEY LEE but not for DERRICK ROSE. As a future study, it is necessary to develop a method that can handle such complicated relations.

6.5 Use of writer information

We first look at the results of an extension of Puduppully et al. (2019)'s model with writer information $\boldsymbol{w}$ in Table 4. By adding $\boldsymbol{w}$ to content planning (stage 1), the method obtained improvements in CS (37.60 to 47.25), CO (16.97 to 22.16), and BLEU score (13.96 to 18.18). By adding $\boldsymbol{w}$ to the component for surface realization (stage 2), the method obtained an improvement in BLEU score (13.96 to 17.81), while the effects on the other metrics were not very significant. By adding $\boldsymbol{w}$ to both stages, the method scored the highest BLEU, while the other metrics were not very different from those obtained by adding $\boldsymbol{w}$ to stage 1. This result suggests that writer information contributes to both content planning and surface realization when it is properly used, and improvements of content planning lead to much better performance in surface realization.

Our model showed improvements in most metrics and showed the best performance by incor-

Method	RG		CS			CO	BLEU
	#	P%	P%	R%	F1%	DLD%	
Puduppully et al. (2019)	33.06	83.17	33.06	43.59	37.60	16.97	13.96
+ w in stage 1	28.43	**84.75**	**45.00**	**49.73**	**47.25**	22.16	18.18
+ w in stage 2	35.06	80.51	31.10	45.28	36.87	16.38	17.81
+ w in stage 1 & 2	28.00	82.27	44.37	48.71	46.44	**22.41**	**18.90**
PROPOSED	39.05	**94.38**	35.77	52.05	42.40	19.38	16.15
+ w	30.25	92.00	**50.75**	**59.03**	**54.58**	**25.75**	**20.84**

Table 4: Effects of writer information. w indicates that WRITER embeddings are used. Numbers in **bold** are the largest among the variants of each method.

The **Milwaukee Bucks** defeated the **New York Knicks**, **105-104**, at Madison Square Garden on Wednesday evening. The **Bucks** (**18**-**16**) have been one of the hottest teams in the league, having won five of their last six games, and they have now won six of their last eight games. The **Knicks** (**16**-**19**) have now won six of their last six games, as they continue to battle for the eighth and final playoff spot in the Eastern Conference. **Giannis Antetokounmpo** led the way for Milwaukee, as he tallied **27** points, **13** rebounds, **four** assists, **three** blocked shots and **one** steal, in **39** minutes . **Jabari Parker** added **15** points, **four** rebounds, **three** assists, **one** steal and **one** block, and **6**-of-**8** from long range. **John Henson** added **two** points, **two** rebounds, **one** assist, **three** steals and **one** block. **John Henson** was the only other player to score in double digits for the Knicks, with **15** points, **four** assists, **three** rebounds and **one** steal, in **33** minutes. The Bucks were led by **Derrick Rose**, who tallied **15** points, **four** assists, **three** rebounds and **one** steal in **33** minutes. **Willy Hernangomez** started in place of Porzingis and finished with **15** points, **four** assists, **three** rebounds and **one** steal in **33** minutes. **Willy Hernangomez** started in place of Jose Calderon (knee) and responded with **one** rebound and **one** block. The Knicks were led by their starting backcourt of **Carmelo Anthony** and **Carmelo Anthony**, but combined for just **13** points on 5-of-16 shooting. The Bucks next head to Philadelphia to take on the Sixers on Friday night, while the Knicks remain home to face the Los Angeles Clippers on Wednesday.

(a) Puduppully et al. (2019)

The **Milwaukee Bucks** defeated the **New York Knicks**, **105-104**, at Madison Square Garden on Saturday. The **Bucks** (**18-16**) checked in to Saturday's contest with a well, outscoring the **Knicks** (**16-19**) by a margin of 39-19 in the first quarter. However, New York by just a 25-foot lead at the end of the first quarter, the Bucks were able to pull away, as they outscored the Knicks by a 59-46 margin into the second. 45 points in the third quarter to seal the win for New York with the rest of the starters to seal the win. The Knicks were led by **Giannis Antetokounmpo**, who tallied a game-high **27** points, to go along with **13** rebounds, **four** assists, **three** blocks and a steal. The game was a crucial night for the Bucks' starting five, as the duo was the most effective shooters, as they posted Milwaukee to go on a pair of low low-wise (Carmelo Anthony) and Malcolm Brogdon. **Anthony** added **11** rebounds, **seven** assists and **two** steals to his team-high scoring total. **Jabari Parker** was right behind him with **15** points, **four** rebounds, **three** assists and a block. **Greg Monroe** was next with a bench-leading **18** points, along with **nine** rebounds, **four** assists and **three** steals. **Brogdon** posted **12** points, **eight** assists, **six** rebounds and a steal. **Derrick Rose** and **Courtney Lee** were next with a pair of {**11** / **11**} -point efforts. **Rose** also supplied **four** assists and **three** rebounds, while **Lee** complemented his scoring with **three** assists, a rebound and a steal. **John Henson** and **Mirza Teletovic** were next with a pair of {**two** / **two**} -point efforts. **Teletovic** also registered **13** points, and he added a rebound and an assist. **Jason Terry** supplied **eight** points, **three** rebounds and a pair of steals. The Cavs remain in last place in the Eastern Conference's Atlantic Division. They now head home to face the Toronto Raptors on Saturday night.

(b) Our model

Table 5: Example summaries generated with Puduppully et al. (2019)'s model (left) and our model (right). Names in **bold face** are salient entities. **Blue numbers** are correct relations derived from input data records but are not observed in reference summary. **Orange numbers** are incorrect relations. **Green numbers** are correct relations mentioned in reference summary.

porating writer information w. As discussed in Section 4.5, w is supposed to affect both content planning and surface realization. Our experimental result is consistent with the discussion.

7 Conclusion

In this research, we proposed a new data-to-text model that produces a summary text while tracking the salient information that imitates a human-writing process. As a result, our model outperformed the existing models in all evaluation measures. We also explored the effects of incorporating writer information to data-to-text models. With writer information, our model successfully generated highest quality summaries that scored 20.84 points of BLEU score.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions. This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), JST PRESTO (Grant Number JPMJPR1655), and AIST-Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory (RWBC-OIL).

References

Tatsuya Aoki, Akira Miyazawa, Tatsuya Ishigaki, Keiichi Goshima, Kasumi Aoki, Ichiro Kobayashi, Hiroya Takamura, and Yusuke Miyao. 2018. Generating Market Comments Referring to External Resources. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 135–139.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*.

Regina Barzilay and Mirella Lapata. 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*, pages 331–338.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating Action Dynamics with Neural Process Networks. In *Proceedings of the Sixth International Conference on Learning Representations*.

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*, pages 128–135.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural Text Generation in Stories Using Entity Representations as Context. In *Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2250–2260.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149.

Luong Hoang, Sam Wiseman, and Alexander Rush. 2018. Entity Tracking Improves Cloze-style Reading Comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1055.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic Entity Representations in Neural Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839.

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.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 850–855.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

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*, pages 91–99.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text Generation by Structure-aware Seq2seq Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730.

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*, pages 1374–1384.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

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*, pages 311–318.

Karl Pearson. 1901. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-Text Generation with Content Selection and Planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *Proceedings of the Sixth International Conference on Learning Representations*.

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*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 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*, pages 1282–1288.

Yasufumi Taniguchi, Yukun Feng, Hiroya Takamura, and Manabu Okumura. 2019. Generating Live Soccer-Match Commentary from Play Data. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *Proceedings of the Third International Conference on Learning Representations*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-Aware Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859.

A Algorithm

The generation process of our model is shown in Algorithm 1. For a concise description, we omit the condition for each probability notation. <SoD> and <EoD> represent "start of the document" and "end of the document", respectively.

B Experimental settings

We set the dimensions of the embeddings to 128, and those of the hidden state of RNN to 512 and all of parameters are initialized with the Xavier initialization (Glorot and Bengio, 2010). We set the maximum number of epochs to 30, and choose the model with the highest BLEU score on the development data. The initial learning rate is 2e-3 and AMSGrad is also used for automatically adjusting the learning rate (Reddi et al., 2018). Our implementation uses DyNet (Neubig et al., 2017).

Algorithm 1: Generation process

```
Input: Data records $\boldsymbol{s}$,
Annotations $Z_{1:T}, E_{1:T}, A_{1:T}, N_{1:T}$
1  Initialize $\{\boldsymbol{r}_{e,a,v}\}_{r \in \boldsymbol{x}}$, $\{\bar{\boldsymbol{e}}\}_{e \in \mathcal{E}}$, $\boldsymbol{h}_0^{\text{LM}}$, $\boldsymbol{h}_0^{\text{ENT}}$
2  $t \leftarrow 0$
3  $e_t, y_t \leftarrow$ NONE, $<$ SOD $>$
4  while $y_t \neq <$ EOD $>$ do
5      $t \leftarrow t+1$
6      if $p(Z_t = 1) \geq 0.5$ then
           /* Select the entity                                                */
7          $e_t \leftarrow \arg\max p(E_t = e'_t)$
8          if $e_t \notin \mathcal{E}_{t-1}$ then
               /* If $e_t$ is a new entity                                      */
9              $\boldsymbol{h}_t^{\text{ENT}'} \leftarrow \text{GRU}^{\text{E}}(\bar{\boldsymbol{e}}_t, \boldsymbol{h}_{t-1}^{\text{ENT}})$
10             $\mathcal{E}_t \leftarrow \mathcal{E}_{t-1} \cup \{e_t\}$
11         else if $e_t \neq e_{t-1}$ then
               /* If $e_t$ has been observed before, but is different from
                  the previous one.                                             */
12             $\boldsymbol{h}_t^{\text{ENT'}} \leftarrow \text{GRU}^{\text{E}}(\boldsymbol{W}^S \boldsymbol{h}_s^{\text{ENT}}, \boldsymbol{h}_{t-1}^{\text{ENT}})$,
13             where $s = \max\{s : s \leq t-1, e = e_s\}$
14         else
15             $\boldsymbol{h}_t^{\text{ENT'}} \leftarrow \boldsymbol{h}_{t-1}^{\text{ENT}}$
           /* Select an attribute for the entity, $e_t$.                        */
16         $a_t \leftarrow \arg\max p(A_t = a'_t)$
17         $\boldsymbol{h}_t^{\text{ENT}} \leftarrow \text{GRU}^{\text{A}}(\boldsymbol{r}_{e_t, a_t, \boldsymbol{x}[e_t, a_t]}, \boldsymbol{h}_t^{\text{ENT}'})$
18         if $a_t$ is a number attribute then
19             if $p(N_t = 1) \geq 0.5$ then
20                 $y_t \leftarrow$ numeral of $\boldsymbol{x}[e_t, a_t]$
21             else
22                 $y_t \leftarrow \boldsymbol{x}[e_t, a_t]$
23             end
24         else
25             $y_t \leftarrow \boldsymbol{x}[e_t, a_t]$
26         $\boldsymbol{h}'_t \leftarrow \tanh\left(\boldsymbol{W}^H(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}})\right)$
27         $\boldsymbol{h}_t^{\text{LM}} \leftarrow \text{LSTM}(\boldsymbol{y}_t \oplus \boldsymbol{h}'_t, \boldsymbol{h}_{t-1}^{\text{LM}})$
28     else
29         $e_t, a_t, \boldsymbol{h}_t^{\text{ENT}} \leftarrow e_{t-1}, a_{t-1}, \boldsymbol{h}_{t-1}^{\text{ENT}}$
30         $\boldsymbol{h}'_t \leftarrow \tanh\left(\boldsymbol{W}^H(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}})\right)$
31         $y_t \leftarrow \arg\max p(Y_t)$
32         $\boldsymbol{h}_t^{\text{LM}} \leftarrow \text{LSTM}(\boldsymbol{y}_t \oplus \boldsymbol{h}'_t, \boldsymbol{h}_{t-1}^{\text{LM}})$
33     end
34     if $y_t$ is "." then
35         $\boldsymbol{h}_t^{\text{ENT}} \leftarrow \text{GRU}^{\text{A}}(\boldsymbol{v}_{\text{REFRESH}}, \boldsymbol{h}_t^{\text{ENT}})$
36 end
37 return $y_{1:t-1}$;
```