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 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

1We 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 ROTO WIRE 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 ROTO WIRE, 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 ROTO WIRE 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 ROTO WIRE 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
Jabari Parker contributed 15 points, four rebounds, three assists. 

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

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

1. $Z_t$: binary variable that determines whether the model refers to input $x$ 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.
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 \( r \in \mathcal{X} \) consists of entity \( e \in \mathcal{E} \), attribute \( a \in \mathcal{A} \), and its value \( x[e,a] \), and is therefore represented as \( r = (e,a,x[e,a]) \). For example, the box-score in Table 1 has a record \( r \) such that \( e = \text{ANTHONY DAVIS}, a = \text{PTS}, \) and \( x[e,a] = 20 \).

### 4.1 Initialization

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

Given the embedding of entity \( e \), attribute \( a \), and its value \( v \), 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 \( r_{e,a,v} \) as follows:

\[
r_{e,a,v} = \tanh \left( W^R (e \oplus a \oplus v) \right),
\]

where \( \oplus \) indicates the concatenation of vectors, and \( W^R \) denotes a weight matrix.\(^3\)

We obtain \( \bar{e} \) in the set of data records \( \mathcal{X} \), by summing all the data-record embeddings transformed by a matrix:

\[
\bar{e} = \tanh \left( \sum_{a \in \mathcal{A}} W^A r_{e,a,x[e,a]} \right),
\]

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

To initialize the hidden state of each module, we use embeddings of \(<\text{SoD}>\) for \( h^{LM} \) and averaged embeddings of \( \bar{e} \) for \( h^{ENT} \).

### 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:

\[
p(Z_t = 1 | h^{LM}_{t-1}, h^{ENT}_{t-1}) = \sigma(W_z (h^{LM}_{t-1} \oplus h^{ENT}_{t-1})),
\]

where \( \sigma(\cdot) \) is the sigmoid function. If \( p(Z_t = 1 | h^{LM}_{t-1}, h^{ENT}_{t-1}) \) 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 \( h^{ENT}_t = h^{ENT}_{t-1} \) (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 \( h^{ENT} \). 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:

\[
p(E_t = e | h^{LM}_{t-1}, h^{ENT}_{t-1}) \propto \begin{cases} 
\exp \left( h^{ENS}_s W^{OLD} h^{LM}_t \right) & \text{if } e \in \mathcal{E}_{t-1} \\
\exp \left( \bar{e} W^{NEW} h^{LM}_t \right) & \text{otherwise}
\end{cases},
\]

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 \( s = \max \{ s : s \leq t - 1, e = e_s \} \), 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 \( h^{ENT} \) with a recurrent neural network with a gated recurrent unit (GRU; Chung et al., 2014):

\[
h^{ENT'}_t = \begin{cases} 
h^{ENT}_{t-1} & \text{if } e_t = e_{t-1} \\
\text{GRU}^E(\bar{e}, h^{ENT}_{t-1}) & \text{else if } e_t \notin \mathcal{E}_{t-1} \\
\text{GRU}^E(W^A \bar{e}, h^{ENT}_{t-1}) & \text{otherwise}
\end{cases}
\]

\(^3\)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 (\( e_t \not\in \mathcal{E}_{t-1} \)), the hidden state of the tracking model is updated with the embedding \( \bar{e} \) 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 (\( e_t \not\equiv e_{t-1} \)), we use \( h^\text{ENT}_t \) (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 \( h^\text{ENT}_t \), we next select the attribute of the salient entity by the following probability:

\[
p(A_t = a \mid e_t, h^\text{LM}_{t-1}, h^\text{ENT}_t) \propto \exp \left( r_{e_t,a,x[e_t,a_t]} W^\text{ATTR}(h^\text{LM}_{t-1} \oplus h^\text{ENT}_t) \right).
\]

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

\[
h^\text{ENT}_t = \text{GRU}^A(r_{e_t,a_t,x[e_t,a_t]}, h^\text{ENT}_{t-1}).
\]

### 4.4 Summary generation

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

If the model refers to a new data record (\( Z_t = 1 \)), it directly copies the value of the data record \( x[e_t,a_t] \). 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., \( \text{fourteen} \)). We decide which one to use by the following probability:

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

where \( W^N \) is a weight matrix. The model then updates the hidden states of the language model:

\[
h^\prime_t = \tanh \left( W^H(h^\text{LM}_{t-1} \oplus h^\text{ENT}_t) \right),
\]

where \( W^H \) 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 \( h^\prime_t \):

\[
p(y_t \mid h^\prime_t) = \text{softmax}(W^Y h^\prime_t).
\]

Subsequently, the hidden state of language model \( h^\text{LM}_t \) is updated:

\[
h^\text{LM}_t = \text{LSTM}(y_t \oplus h^\prime_t, h^\text{LM}_{t-1}),
\]

where \( y_t \) is the embedding of the word generated at time step \( t \).

### 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 \( \bar{w} \) of a writer to \( h^\text{LM}_{t-1} \oplus h^\text{ENT}_t \) to construct context vector \( h^\prime_t \):

\[
h^\prime_t = \tanh \left( W^{R/H}(h^\text{LM}_{t-1} \oplus h^\text{ENT}_t \oplus \bar{w}) \right),
\]

where \( W^{R/H} \) is a new weight matrix. Since this new context vector \( h^\prime_t \) 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 \( h^\prime_t \) enhanced with the writer information is used to obtain \( h^\text{LM}_t \), 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:

\[
\log p(Y_1:T, Z_1:T, E_1:T, A_1:T, N_1:T \mid x) = \sum_{t=1}^{T} \log p(Z_t = z_t \mid h^\text{LM}_{t-1}, h^\text{ENT}_t) + \sum_{t:Z_t=1} \log p(E_t = e_t \mid h^\text{LM}_{t-1}, h^\text{ENT}_t) + \sum_{t:Z_t=1} \log p(A_t = a_t \mid e_t, h^\text{LM}_{t-1}, h^\text{ENT}_t) + \sum_{t:Z_t=1} \log p(N_t = n_t \mid h^\text{LM}_{t-1}, h^\text{ENT}_t) + \sum_{t:Z_t=0} \log p(Y_t = y_t \mid h^\prime_t)
\]

\[\text{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 \( \text{REFRESH} \) to refresh these states after our model generates a period:} \]

\[h^\text{ENT}_t = \text{GRU}^A(\text{REFRESH}, h^\text{ENT}_{t-1})\]
Table 3: Experimental result. Each metric evaluates whether important information (CS) is described accurately (RG) and in correct order (CO).

| Method            | RG # | P% | CS P% | R% | F1% | CO DLD% | BLEU |
|-------------------|------|----|-------|----|-----|---------|------|
| GOLD Templates    | 27.36| 93.42 | 100.   | 100.   | 100. | 100.   | 100.   |
| 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** |

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 $r \in x$, 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 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 $h^{E_N}_t$ 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 $w$ with the input vector for LSTM in the content planning decoder (stage 1);
2. concatenating writer embedding $w$ 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:

$$\text{BLEU} = \frac{\text{log}(n)}{\text{log}(n + k)}$$

$$\text{BLEU} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{log}(n_i)}{\text{log}(n_i + k)}$$

$$\text{BLEU} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{log}(n_i)}{\text{log}(n_i + k)}$$

$$\text{BLEU} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{log}(n_i)}{\text{log}(n_i + k)}$$

$$\text{BLEU} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{log}(n_i)}{\text{log}(n_i + k)}$$

where $n_i$ is the number of words in the $i^{th}$ translation and $k$ is the length of the $i^{th}$ reference.

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-
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. 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 $\bar{e}$, which depends on the box score for each game in addition to static entity embedding $e$. 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 $e$, the top-ranked players are closely located.

We also present the visualizations of dynamic entity embeddings $\bar{e}$ 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) for representing the dynamic entity embedding $\bar{e}$, 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.

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-
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 COURTFIELD 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 \( w \) in Table 4. By adding \( w \) 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 \( w \) 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 \( w \) to both stages, the method scored the highest BLEU, while the other metrics were not very different from those obtained by adding \( w \) 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-
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.

| Method                | RG \( # \) | P\% | P\% | CS \( R\% \) | F1\% | CO \( DLD\% \) | BLEU |
|-----------------------|------------|-----|-----|--------------|-----|-----------|------|
| 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 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.

(a) Puduppully et al. (2019)  
(b) Our model

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 Martenschat, 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.
The generation process of our model is shown in Algorithm 1. For a concise description, we omit the condition for each probability notation. \texttt{<SOD>} and \texttt{<EOD>} represent “start of the document” and “end of the document”, respectively.

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. \texttt{<SOD>} and \texttt{<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 \( s \), Annotations \( Z_{t:T}, E_{t:T}, A_{t:T}, N_{t:T} \)

1. Initialize \( \{ r_{e,a,v} \}_{r \in \mathbb{R}}, \{ \bar{e} \}_{e \in E}, h_{0}^{LM}, h_{0}^{ENT} \)
2. \( t \leftarrow 0 \)
3. \( e_{t}, y_{t} \leftarrow \text{NONE}, \text{<SOD>} \)

4. while \( y_{t} \neq \text{<EOD>} \) do

5. \( t \leftarrow t + 1 \)

6. if \( p(Z_{t} = 1) \geq 0.5 \) then

7. /* Select the entity */

8. \( e_{t} \leftarrow \text{arg max} p(E_{t} = e_{t}) \)

9. if \( e_{t} \notin \mathcal{E}_{t-1} \) then

10. /* If \( e_{t} \) is a new entity */

11. \( h_{t}^{ENT} \leftarrow \text{GRU}^{E}(\bar{e}_{t}, h_{t-1}^{ENT}) \)

12. \( \mathcal{E}_{t} \leftarrow \mathcal{E}_{t-1} \cup \{ e_{t} \} \)

13. else if \( e_{t} \neq e_{t-1} \) then

14. /* If \( e_{t} \) has been observed before, but is different from the previous one. */

15. \( h_{t}^{ENT} \leftarrow \text{GRU}^{E}(W S h_{s}^{ENT}, h_{t-1}^{ENT}), \)

16. where \( s = \max \{ s : s \leq t-1, e = e_{s} \} \)

17. else

18. /* Select an attribute for the entity, \( e_{t} \). */

19. \( a_{t} \leftarrow \text{arg max} p(A_{t} = a_{t}') \)

20. \( h_{t}^{ENT} \leftarrow \text{GRU}^{A}(r_{e_{t}, a_{t}, x[e_{t}, a_{t}]}, h_{t}^{ENT}) \)

21. if \( a_{t} \) is a number attribute then

22. if \( p(N_{t} = 1) \geq 0.5 \) then

23. \( y_{t} \leftarrow \text{numeral of} \ x[e_{t}, a_{t}] \)

24. else

25. \( y_{t} \leftarrow x[e_{t}, a_{t}] \)

26. end

27. else

28. \( y_{t} \leftarrow x[e_{t}, a_{t}] \)

29. \( h_{t}^{L} \leftarrow \text{tanh} (W H(h_{t-1}^{LM} \oplus h_{t}^{ENT})) \)

30. \( h_{t}^{LM} \leftarrow \text{LSTM}(y_{t} \oplus h_{t}^{L}, h_{t-1}^{LM}) \)

31. else

32. \( e_{t}, a_{t}, h_{t}^{ENT} \leftarrow e_{t-1}, a_{t-1}, h_{t-1}^{ENT} \)

33. \( h_{t}^{L} \leftarrow \text{tanh} (W H(h_{t-1}^{LM} \oplus h_{t}^{ENT})) \)

34. \( y_{t} \leftarrow \text{arg max} p(Y_{t}) \)

35. end

36. if \( y_{t} \) is “.” then

37. \( h_{t}^{ENT} \leftarrow \text{GRU}^{A}(v_{\text{REFRESH}}, h_{t}^{ENT}) \)

38. end

39. return \( y_{1:t-1} \);