Natural Language Engineering (2024), **30**, pp. 429–453
doi:10.1017/S135132492300013X

ARTICLE

SAN-T2T: An automated table-to-text generator based on selective attention network

Haijie Ding[1] and Xiaolong Xu[2]

[1]Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, China and [2]School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China
Corresponding author: Xiaolong Xu; Email: xuxl@njupt.edu.cn

(Received 3 February 2021; revised 3 November 2022; accepted 7 April 2023; first published online 5 May 2023)

Abstract

Table-to-text generation aims to generate descriptions for structured data (i.e., tables) and has been applied in many fields like question-answering systems and search engines. Current approaches mostly use neural language models to learn alignment between output and input based on the attention mechanisms, which are still flawed by the gradual weakening of attention when processing long texts and the inability to utilize the records' structural information. To solve these problems, we propose a novel generative model SAN-T2T, which consists of a field-content selective encoder and a descriptive decoder, connected with a selective attention network. In the encoding phase, the table's structure is integrated into its field representation, and a content selector with self-aligned gates is applied to take advantage of the fact that different records can determine each other's importance. In the decoding phase, the content selector's semantic information enhances the alignment between description and records, and a featured copy mechanism is applied to solve the rare word problem. Experiments on WikiBio and WeatherGov datasets show that SAN-T2T outperforms the baselines by a large margin, and the content selector indeed improves the model's performance.

Keywords: neural language; selective attention network; semantic vector; table-to-text

1. Introduction

Generating descriptions for structured table data is an essential task for natural language generation, aiming to take table data as input to generate text that adequately and coherently describes these data (Parikh *et al.*, 2020). It has been applied in many domains, typically including weather forecasting, question-answering systems, sports event broadcasts, and biographical writings. For example, a summary text needs to be generated in a sports event, based on real-time and detailed game data, to present to the audiences quickly.

There are some differences between table-to-text generation and the broader task of text generation (e.g., machine translation; Zhang, Xiong, and Su 2020) and usually need to solve two levels of problems (Wiseman, Shieber, and Rush, 2017):

1. Select the appropriate subset of data to describe. Unlike general text sequences, tables often contain additional information such as fields, values (linguistic and structural), and table titles. For example, when writing a biography for a person based on the data, we will firstly determine which ones to be included from a large amount of confusing and trivial information.

2. How to present comprehensive and precise information based on refined semantic information. In a table where only a few of the vast records are suitable for the final output, generating coherent and relevant text is the ultimate goal of this task. The second problem lies in arranging the order of them correctly. Understanding these latent semantic structures and then refining them is the foundation for the table-to-text generation.

The deep learning systems based on the neural network language models (NLM; LeCun *et al.*, 2015) blur the boundaries between these two processes and modularly handle both subtasks through an end-to-end system (Zhang, Zhang, and Yan, 2019). However, Wiseman *et al.* (2017) find that although the NLM-based approaches can generate descriptions fluently, they are still not as accurate as the template-based systems in selecting the subset of records and yet have room for improvement. Current neural language models proposed to solve table-to-text generation mainly rely on the encoder–decoder structure (Sutskever, Vinyals, and Le, 2014) and attention mechanism (Bahdanau, Cho, and Bengio, 2014). Mei *et al.* (2016) propose a selective-generation method with an encoder–decoder-aligner structure to generate weather forecasts, using a pre-defined corrector model to correct the dependencies between the records and the text. Lebret *et al.* (2016) model the fields and local and global conditioning in the WikiBio dataset based on an n-gram model and improve the effect. Liu *et al.* (2018) use dual attention and a flexible copy to solve the dependencies between records and text, and finally improve the effect by about 10 points than the baseline. Puduppully *et al.* (2019) improve the RotoWire dataset's outcome by modeling the generation as content planning and surface realization. Bao *et al.* (2019) have made some improvements on WikiBio using the encoder–decoder structure and conditional copy mechanism for handling text alignment and rare word problems; they propose a new multi-line tabular dataset WikiTableText. These works have inspired our work. However, there are still some problems to be solved: tables often contain complex structural information, and the alignment from the attention vector would gradually diminish in generating long text, so the content of the records cannot be extracted well (Chen *et al.*, 2020a, 2020d).

To this end, we propose a novel generative model based on the encoder–decoder framework and attention mechanism, SAN-T2T, using long short-term memory network (LSTM; Hochreiter and Schmidhuber, 1997) to process the complicated structure in the table. In the encoding phase, we integrate the table's structure into its representation and add a position gate to the cell state of the LSTM unit. Then we apply the self-gated content selector to utilize its reciprocal inference (i.e., different records may determine each other's importance in the table's structure) and an LSTM decoder to generate the description. In the decoding phase, the semantic vector from the content selector will enhance the alignment between the output and input, and a featured copy mechanism is applied to alleviate the rare word problem.

Our contributions are as follows:

1. We utilize the fields' position to enhance the decoder's ability to extract structural information from the attention mechanism.
2. The gate mechanism is used to determine the importance of mutual decision-making between different fields in advance. The modified attention vector determines the amount of information that each timestep in the decoder obtains from the source sequence.
3. In the inference phase, the most relevant words are copied from the source sequence based solely on the attention vector to alleviate the problem of rare words.
4. Finally, we also experimentally compare the effect of beam search on the model's performance.

We have performed experiments on SAN-T2T with the Bilingual Evaluation Understudy (BLEU) and recall-oriented understudy for Gisting evaluation (ROUGE) metrics. Results and attention visualizations show that SAN-T2T can precisely understand the table's content and structure and then generate comprehensive and correct descriptive text.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

2. Related work

Table-to-text generation can usually be divided into two independent modules (Wiseman *et al.*, 2017):

1. Content selection (CS). Select an appropriate subset of related records to be described from the source table data. Each record has a wide range of complicated and aggregate statistics, with only a small subset of this information having a positive influence on the description, the CS component is to choose the subset which works properly.
2. Surface realization. Generate descriptions for these selected subsets. For the chosen records from the CS component, semantic information along with other features should be obtained precisely.

Many approaches have been proposed to solve these two problems. There are some typical methods for CS, such as Barzilay and Lapata (2005) to build this by aligning the records and text. They treat CS as a collective classification problem and build links between entities with similar labels to consider all candidates simultaneously rather than select each item separately. Surface realization is usually regarded as obtaining semantic information from the given feature representation and then generating text. Reiter and Dale (2000) use various language feature models and hand-built grammars to design text generators for surface realization. On top of this model, Reiter (2007) then extended his architecture so that the input of the system was raw data rather than artificial intelligence knowledge base, and the task was divided into four stages, which were processed successively. These methods are mainly based on template rules or statistical language models to solve the alignments between text and records, while most of the recent works combine CS and surface realization in a unified end-to-end framework (Kondadadi, Howald, and Schilder, 2013; Konstas and Lapata, 2013a; Oya *et al.*, 2014). But it can be considered as a neural extension of the Probabilistic Context-Free Grammar (PCFG) system in Konstas and Lapata (2013b), with a more powerful transition probability considering inter-segment dependence and a state-of-the-art attention-based language model as the linguistic realizer. Wiseman *et al.* (2018) integrate the end-to-end model and the rules-based template system, using a hidden semi-Markov model which is suitable for fragment modeling to learn to extract and use templates, then parameterize these probabilities with a neural language model, and finally use the Viterbi algorithm to infer the hidden states and use them as the template. Jiang *et al.* (2020) also integrate these two modules and propose the pipeline-assisted neural networks to conduct table-to-text generation tasks in social Internet of Things. They analyze records correlation and filter redundant records to make full use of attention and gate mechanism and to improve CS.

The above-mentioned works all model the table-to-text task on the encoder–decoder framework, which encodes a source sequence into a fixed-length vector from which a decoder generates the target sequence. The encoder–decoder models aim to solve the problem that previous deep neural networks cannot handle most sequential problems (e.g., machine translation) because they can only be applied to solve problems which have fixed-dimensional inputs and targets. Sutskever *et al.* (2014) propose the encoder–decoder model which now performs as the baseline for many sequence-to-sequence problems. They use the LSTM network to encode the input sequence, one timestep at a time, to obtain a fixed-dimensional vector representation and then use another LSTM network to extract semantic information from this vector. LSTM network doing well at learning dependencies from long-range sequences makes it the first choice for the sequence-to-sequence problems because of the considerable time lags between the inputs and outputs. Though the great advances in the encoder–decoder framework, there are several shortages either. It encodes the source sequence into a fixed-length hidden semantic vector, which leads to two problems. First, it cannot compress the full necessary information into the fixed-length hidden vector. Second, the previously hidden information will be covered due to the gates in the LSTM unit, making it harder to cope with sequential problems with long sentences, especially those much longer than

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

the average length of the training corpus. To solve this problem, Bahdanau et al. propose the attention mechanism in 2014 which is now widely used for natural language processing (NLP) and then explore the promising applications of the encoder–decoder architecture. The attention models can automatically (soft)-align parts of the source sequence that are relevant to predicting the target sequence, without having to form these parts as a hard segment explicitly. This is done by weighing the relevance (alignment) of any entities of the input text and taking such a weight into account when predicting the result. The idea is built on the hypothesis that background features could be irrelevant regarding some objects in the foreground, but relevant to others considering the context. However, such attention approaches are not always reliable; when processing long or information-complicated sequences, the attention context representation for alignment would gradually be weakened.

Based on the previous works, the bottleneck of current models for table-to-text tasks lies in the stage of content planning (i.e., select and order salient content from the input (Gong *et al.*, 2020). The SAN-T2T model borrows the idea of representing hidden features through the fields and their corresponding content to describe the structured tables more accurately (Lebret *et al.*, 2016). However, the n-gram model proposed by Lebret et al. cannot effectively model the distant dependencies in the sequence. Mei et al. propose a Seq2seq model in 2016 which aligns dependencies between records and text through a pre-defined corrector and use one-hot vectors to represent features of the records, but due to the limitations of one-hot encoding, their model could not represent tables with complex structures, such as infoboxes in the WikiBio dataset. Li *et al.* (2019) present a transformer-based generation model, modify the latent representation of the record embedding, and propose two data augmentation methods. To improve performance in CS and coherent ordering, Puduppully and Lapata (2021) propose a neural model with a macro planning stage followed by a generation stage reminiscent of traditional methods which embrace separate modules for planning and surface realization.

To address the problem of out-of-vocabulary (OOV) words appearing in the records (including some special entities, names, etc.), See *et al.* (2017) and Gu *et al.* (2016) design the copy mechanisms. When generating certain special terms, the OOV words could be copied from the source sequence to the target text. Sha *et al.* (2018) design a linking mechanism with link-based and content-based attention to model the content order, a self-adaptive gate to balance these two levels of attention, and then utilize a copy network to solve the OOV problem. Liu *et al.* (2018) extend the work of Sha et al., by using dual attention to align dependencies between records and text, and the word-level attention is focused on the alignment between the output text and input data, while field level is more concerned with that between the text and the overall structure of the table. Qin *et al.* (2018) establish the dependencies using the method of sequence labeling, regard labeling the semantics of the vocabulary as a hidden variable, and design a hidden semi-Markov model for learning and inference. Their model can learn more latent semantic information while retaining interpretability. Wiseman *et al.* (2017) propose a new extractive evaluation metric and a brand new dataset RotoWire, while using joint and conditional copy mechanisms to handle the OOV problem. Later, Puduppully *et al.* (2019) model the generation task as content planning and surface realization, where content planning is used to find a subset of records that keep important roles in the source data and plan a reasonable order to describe these records. Bao *et al.* (2019) design a model based on the encoder–decoder framework, focusing on using copy network to deal with the problem of rare words, but their models do not solve the problem that the attention mechanism cannot make good utilization of the records' structure. To alleviate the training difficulty and consumption, Jean *et al.* (2015) make use of the attention vectors to track the origins of all target words; Luong *et al.* (2015) use unsupervised alignments as the word dictionary to post-process the translation and replace the unknown with the source words. Then Liu *et al.* (2020) use the pointer-generator network to handle the problem of rare words; their model can also reference the slot-value pairs data and then introduce the slot-attention mechanism and coverage mechanism to calculate the attention score using the attribute sequence and value sequence simultaneously and

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

to alleviate the problem of assigning values to the wrong fields. A common problem with table-to-text models is that they produce text descriptions that do not conform to the tabular information, commonly known as "illusions." To solve the "illusion," Rebuffel *et al.* (2022) proposed a multi-branch-weighted decoder and a word-level labeling process. This word-level tagging process can reduce the failure of word matching process through dependency analysis, based on co-occurrence and sentence structure, while still producing correct labels in complex environments.

The current training of neural language models always requires extremely large-scale corpus, while not all tasks have sufficient-scale and high-quality datasets. Ma *et al.* (2019) extract key facts from the table through the sequence labeling model firstly and then combine these key facts as input and use a Seq2Seq model to convert them into text, which alleviates the requirement of data scale for table-to-text generation. Chen *et al.* (2020c) leverage pre-training and transfer learning to address this issue, which consists of a general knowledge-grounded generation model to generate knowledge-enriched text, and a pre-trained model which can be fine-tuned on various table-to-text generation tasks. Although pre-trained models are widely used in other areas of NLP, there are some problems with using pre-trained models in the table-to-text domain. First, there is a big difference between the language input of the pre-trained model and the input of structured data. Second, table information is not a linear input like natural language (Chen *et al.* 2020b). Table has structured information, but the traditional pre-training model has no corresponding method to understand the structured information of table. Moreover, pre-trained models do not solve the "illusion" problem. To alleviate the above problems, Gong *et al.* (2020) proposed Table GPT, which can train the table-to-text model with few samples. In order to deal with the gap between linear sequences of natural languages and structured data tables, Gong et al. proposed a table conversion module, which used templates to transform structured tables into natural languages. In order to solve the problem of insufficient table structure information extraction, an auxiliary task of table structure reconstruction is proposed in the framework of multi-task learning.

3. Table-to-text generator

We propose a novel generative model, SAN-T2T, based on the seq2seq (Sequence-to-Sequence) learning and neural language model. In this section, we introduce the task of table-to-text generation and then reveal our major ideas on data preprocessing, field-content selective encoder, descriptive decoder, and the featured copy mechanism.

3.1. Task definition

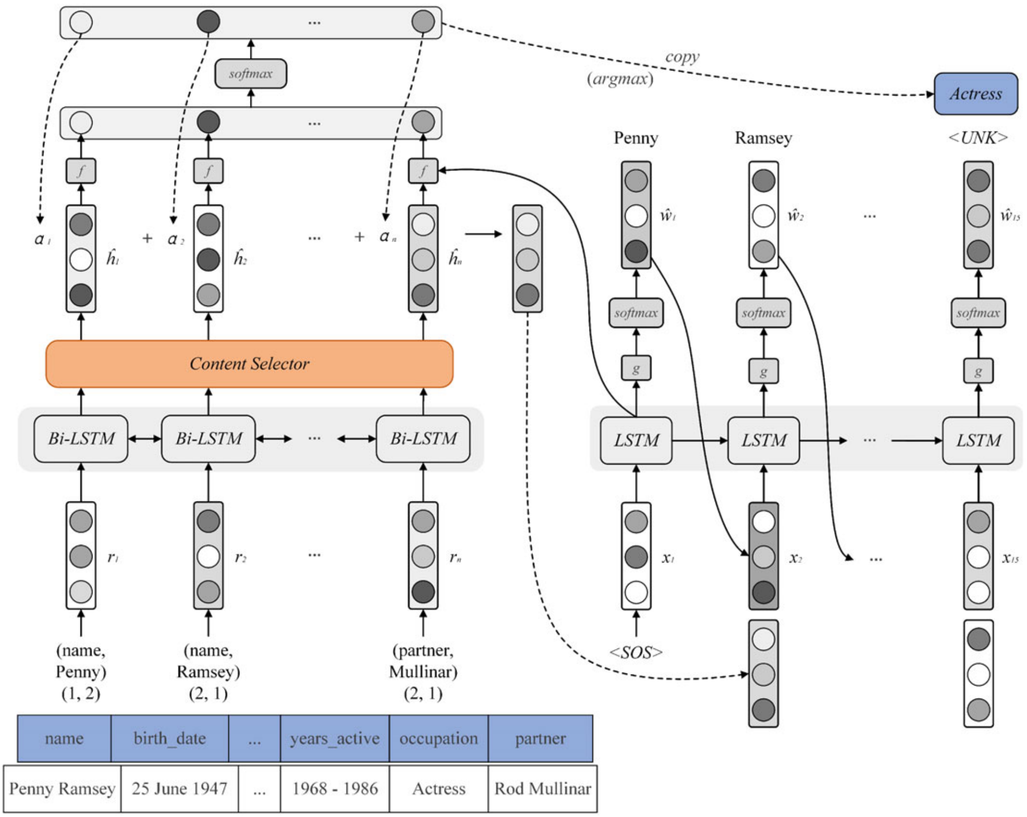
Different from the general seq2seq models to take a sequence as input, SAN-T2T takes a table as input, which consists of an infobox of various records. We model the table-to-text generation as a language model based on the seq2seq structure. The given table T consists of n field-value pairs and their corresponding descriptions. In the training phase, the input and output of the model are the records $r_{1:n}$ in T and the corresponding reference text $w_{1:t}$. In the inference phase, only the input $r_{1:n}$ is given, and the output would be determined by the model. Among them, $w_{1:t}$ contains t words $\{w_1, w_2, w_3, \ldots, w_t\}$, and the purpose of inference is to generate $\widehat{w}_{1:t}$ that maximizes $P(w_{1:t} \mid r_{1:n})$.

$$\widehat{w}_{1:t} = \text{argmax}_{w_{1:t}} \prod_{s=1}^{t} P(w_s \mid w_{0:s-1}, r_{1:n}) \tag{1}$$

We test the SAN-T2T model on three public datasets: WikiBio, RotoWire, and WeatherGov.

WikiBio contains 728,321 Wikipedia biographies. Each sample contains an infobox and the first paragraph of the corresponding biographical text, and each text contains 26.1 words on average.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press



name	birth_date	...	years_active	occupation	partner
Penny Ramsey	25 June 1947	...	1968 - 1986	Actress	Rod Mullinar

Figure 1. Architecture of SAN-T2T.

RotoWire contains 4853 samples, which consist of human-written NBA basketball game summaries aligned with their corresponding box- and line-scores, and each summary text contains 337.1 words on average.

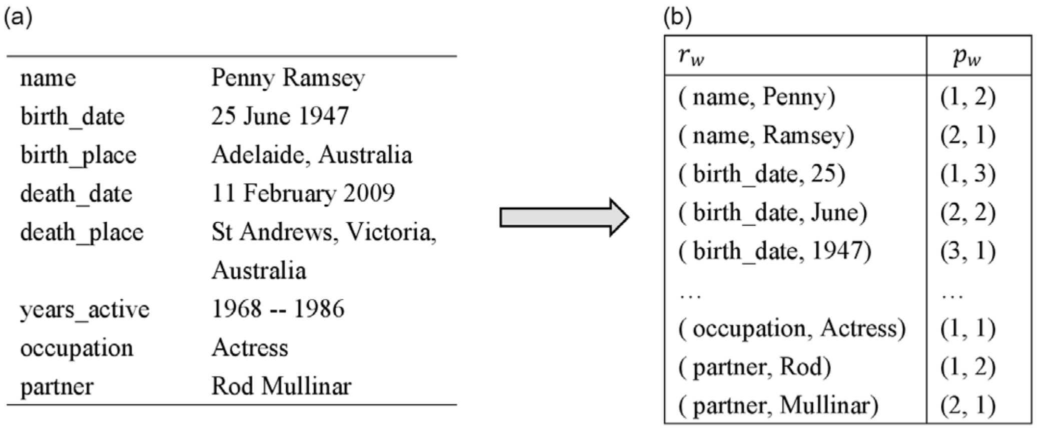
WeatherGov contains 29,528 samples of weather forecast records, and each sample contains 36 records of fixed length, such as temperature and wind speed, and is paired with a description text, which contains 28.7 words on average.

3.2. SAN-T2T

The neural language model SAN-T2T proposed in this paper is based on the selective attention network, which combines gate, attention, and copy mechanism for table-to-text generation. The architecture of SAN-T2T is shown in Figure 1.

SAN-T2T could be introduced through three aspects:

1. Position encoding. In the preprocessing phase, we model the fields' value and location jointly to learn semantic and structural information from the table, as shown in Section 3.3.
2. Field-content selective encoder. We utilize the self-gate mechanism in the encoder to determine the importance of mutual decision-making between different fields. Its architecture is shown in Figure 4. Then attention vector can obtain more information from the source sequence and refine the alignment between the records and text.



(a)

name	Penny Ramsey
birth_date	25 June 1947
birth_place	Adelaide, Australia
death_date	11 February 2009
death_place	St Andrews, Victoria, Australia
years_active	1968 -- 1986
occupation	Actress
partner	Rod Mullinar

The wiki infobox of Penny Ramsey

(b)

r_w	p_w
(name, Penny)	(1, 2)
(name, Ramsey)	(2, 1)
(birth_date, 25)	(1, 3)
(birth_date, June)	(2, 2)
(birth_date, 1947)	(3, 1)
…	…
(occupation, Actress)	(1, 1)
(partner, Rod)	(1, 2)
(partner, Mullinar)	(2, 1)

The preprocessed representation

Figure 2. The wiki infobox of Penny Ramsey and the preprocessed representation.

3. In the inference phase, a formally simple but effective copy mechanism is applied in SAN-T2T that uses attention vectors exclusively rather than training an additional layer of neural networks. This can significantly alleviate the problem of rare words while reducing the resource consumption of additional training.

3.3. Position encoding

Data records in structured tables usually contain vast field-value pairs, where values consist of sequences of words of unfixed length that correspond to the content of this field. The field values are encoded with word embedding, taking the field embedding, which is determined by the embedding vector of the field and its position in the table, as the key information to model the value. Take the WikiBio dataset as an example, Lebret *et al.* (2016) represent field embedding in triples: (f_w, p_w^+, p_w^-), where f_w represents word embedding of the field and p_w^+, p_w^- represent its forward and backward positions, respectively. Inspired by this idea, we take the fields and their appearing positions as auxiliary information for input. As shown in Figure 2, (a) represents the content of Penny Ramsey's Wikipedia infobox, involving her birth and death information and occupation, and (b) is the corresponding preprocessed representation. $r_w = \text{concat}\{f_w, v_w\}$ represents the input, where v_w are the values corresponding to the field. $p_w = \text{concat}\{p_w^+, p_w^-\}$ is the auxiliary position information. r_w and p_w are taken as the joint input of a certain timestep of LSTM unit.

3.4. Field-content selective encoder

The field-content selective encoder contains the field encoder and content selector. The field encoder is designed to encode the records with LSTM network and to get the semantic and structural features of the records. Then the content selector controls the amount of information to be flowed to the decoder, through the gated content-selection algorithm.

3.4.1. Field encoder

The purpose of the field encoder is to encode each input record r_i. LSTM is suitable for processing and predicting related events with long intervals and lags in time series. But unidirectional LSTM

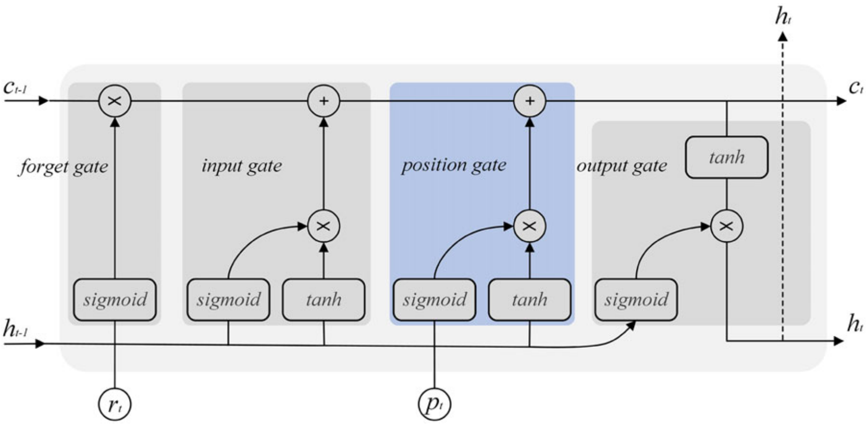


Figure 3. The structure of LSTM unit.

can only predict the next output from the previous states. Therefore, we use an improved bidirectional LSTM to enable it to learn the complex structure and long dependencies of the field-value pairs. Following the design in Graves *et al.* (2013), the structure used in SAN-T2T at each timestep is shown in Figure 3 and defined as:

$$h_t, c_t = \text{biLSTM}\left(r_t, h_{t-1}\right) \tag{2}$$

where $r_t = \text{concat}\{f_t, v_t\}$ is the input at timestep t, and c_t and h_t are the cell state and hidden state at t, respectively. LSTM can reserve essential information through the cell states and hidden states.

As we mentioned above, in order to learn more about the structure of the table, we take $p_t = \text{concat}\{p_t^+, p_t^-\}$ as the joint input of LSTM. The new cell states and hidden states are calculated as:

$$\begin{pmatrix} \varphi_t \\ \omega_t \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{tanh} \end{pmatrix} \left(W_p p_t + b_p\right) \tag{3}$$

$$\acute{c}_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t + \varphi_t \odot \omega_t \tag{4}$$

where $W_p \in \mathbb{R}^{2n\times 2d_p}$, $b_p \in \mathbb{R}^{2n}$ are the weight matrix and bias. $\varphi_t \in (0, 1)^n$ determines the amount of structural information to be stored in c_t, and $\omega_t \in (-1, 1)^n$ retains all the structural information at the current timestep. f_t, i_t, and $\tilde{c}_t$ are obtained from the forget gate and input gate. We believe that the improved LSTM can not only learn the hidden semantic information of the long sequence but also preserve its structural information for the field-value pairs, while the structure information keeps a very important role in the decision-making of the attention mechanism in the decoder.

3.4.2. Content selector with gate mechanism

For the different records in the table, we believe that the context of each record is important in determining the others. For example, if a person is a football player, then the other relevant records, such as his playing position and the team he is playing for, etc., should also appear in the description. Puduppully et al. take these into consideration and apply the CS gating mechanism to obtain the new representation in the encoder. We borrow this idea to better capture the latent dependency as Figure 1, replacing the output of the encoder with that of the content selector so that more accurate semantic information can be obtained for a specific record in the decoder. The details are as Algorithm 1, and Figure 4 illustrates the architecture of the content selector. We

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Algorithm 1. Gated Content-Selection Algorithm

```
Input: Infobox of each sample.
for each field-value pair $(f_i, v_i)$ and positions $(p_i^+, p_i^-)$ in the Infobox do
    Preprocess to $x \leftarrow \{((f_1, v_1), (p_1^+, p_1^-)), \ldots, ((f_n, v_n), (p_n^+, p_n^-))\}$
    Initialize the reciprocal-importance semantic vector list $\acute{h} = []$
    $h \leftarrow \mathrm{LSTM}(x)$
    for each hidden-state $h_i$ in $h$ do
        $\alpha_{i,j} \leftarrow \mathrm{softmax}(h_i W_r h_j)$
        $\tilde{d}_i \leftarrow \mathrm{sigmoid}(W_d \mathrm{concat}\{h_i, \sum_j \alpha_{i,j} h_j\} + b_d)$
        $\acute{h}_i \leftarrow \tilde{d}_i \odot h_i$
    end for
end for
Output: The reciprocal-importance semantic vector list $\acute{h}$
```

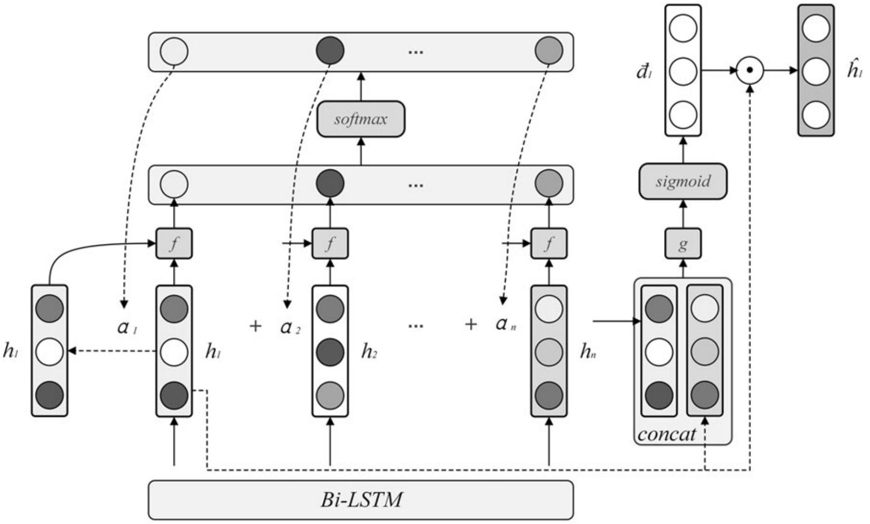


Figure 4. Architecture of the gated content selector.

apply the output of the encoder's LSTM layer to do a self-gate adaption and control the amount of information that flowed to the following decoder component through the sigmoid function. So the input to the decoder will be covered by the content selector's result. Similar to the attention mechanism, we first calculate the importance score of each record $\alpha_{i,j}$, which will be used later to get the dependency vector d_i of this record, and then obtain the new encoder output $\acute{h}_i$ through the content-selection gate.

$$\alpha_{i,j} = \mathrm{softmax}\left(h_i W_r h_j\right) \tag{5}$$

$$d_i = \sum_j \alpha_{i,j} h_j \tag{6}$$

$$\tilde{d}_i = \mathrm{sigmoid}\left(W_d \mathrm{concat}\{h_i, d_i\} + b_d\right) \tag{7}$$

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

$$\acute{h}_i = \tilde{d}_i \odot h_i \tag{8}$$

where $W_r \in \mathbb{R}^{n\times n}$, $W_d \in \mathbb{R}^{n\times 2n}$, and $b_d \in \mathbb{R}^n$ are the learnable weight matrix and bias. The $\acute{d}_i \in (0, 1)^n$ controls the amount of information that can be obtained from the field-content selective encoder at each timestep in the decoder.

3.5. Descriptive decoder

We mainly use LSTM network in the decoder to learn the semantic information of the context. In the training phase, the encoded word embedding x_t at each timestep, the context vector z_t obtained from the selective encoder, and the previous hidden state s_{t-1} are used as input and then output the conditional probability distribution P_t of the next word $P(w_t \mid w_{0:t-1}, r_{1:n})$.

A standard unidirectional and single-layer LSTM network is used in the decoder, and the context vector z_t of each timestep is calculated through the attention mechanism as Bahdanau *et al.* (2014):

$$e_{t,j} = \frac{s_{t-1} \odot \acute{h}_j}{\sqrt{d_n}} \tag{9}$$

$$\alpha_{t,j} = \frac{\exp\left(e_{t,j}\right)}{\sum_j \exp\left(e_{t,j}\right)} \tag{10}$$

$$z_t = \sum_j \alpha_{t,j} \acute{h}_j \tag{11}$$

where s_t, c_t are the hidden state and cell state output by the LSTM network.

We integrate the embedding vector of the target sequence word and context vector as input to the LSTM unit and then do the *softmax* function to calculate each word's generating probability:

$$\acute{x}_t = \text{concat}\{x_t, z_t\} \tag{12}$$

$$s_t, c_t = \text{LSTM}\left(\acute{x}_t, s_{t-1}\right) \tag{13}$$

$$P(w_t \mid w_{0:t-1}, r_{1:n}) = \text{softmax}\left(W_s s_t + b_s\right) \tag{14}$$

$$\widehat{w}_t = \text{argmax}_{w_t} P\left(w_t \mid w_{0:t-1}, r_{1:n}\right) \tag{15}$$

where $W_s \in \mathbb{R}^{d_{out}\times n}$ and $b_s \in \mathbb{R}^{d_{out}}$ are the weight matrix and bias. $\widehat{w}_t$ is the word to be generated at timestep t.

Since the records in some specific datasets (e.g., WeatherGov) are fixed-length rather than variable-length key-value pairs, and there exists no reciprocal location between different records, it does not make much sense to encode each record with the field encoder, respectively. To ensure that the model can still learn the dependency between the output text and records through the content selector without implementing the field encoding, we apply the method proposed by Mei *et al.* (2016), using the regularization-term-added cross entropy as the loss function:

$$\text{Loss} = -\sum_{j=1}^{t} \log \frac{\exp(y\left[w_j\right])}{\sum_j \exp(y[j])} + \left(\sum_{j=1}^{n} cs_j - \gamma\right)^2 + \left(1 - \max\left(cs_j\right)\right) \tag{16}$$

where $cs_j \in (0, 1)$ is the context vector obtained by content selector, which is calculated by the *sigmoid* function, and represents the probability of each record being selected; the prunable

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

parameter γ can force the model to depend on the specific number of records. $(\sum_{j=1}^{n} cs_j - \gamma)^2$ enables the model to correct its weight based on the content selector and the context vector of the selected record through gradient descent, while $(1 - \max(cs_j))$ can ensure that the content selector has selected at least one record.

3.6. Inference model with copy mechanism

In the training phase, the records $r_{1:n}$ and reference text $w_{1:t}$ are both taken as inputs into the encoder and decoder, respectively, been trained to maximize the likelihood of generating the real text $\widehat{w}_{1:t}$. However, in the inference phase, text is generated by finding the words with the largest posterior probability predicted by the trained model. In other words, the subsequent words are generated sequentially from the first word.

We also apply beam search to improve the model performance. The strategy is that the model will find a string of generated words that approximately maximizes the conditional probability given the previous states, so it will generate the text from begin-to-end and keep a fixed number (i.e., beam width) of candidates with the highest log-probability at each timestep.

Meanwhile, to alleviate the problem of rare words, a flexible copy mechanism is used to copy OOV words to the output text from the source sequence. This is done by replacing the <*unk*> character in the output with the most relevant word in the table, which is the field pointed to by the attention vector with *argmax* function. Experiments have proved that the copy mechanism we used can greatly improve the performance of the model without increasing additional consumption. As shown in Fig. 1, the model actually outputs <*unk*> when generating "actress," but by pointing to the most relevant field "occupation" in the table via the attention vector α, the <*unk*> is replaced with its value "actress" accordingly.

The specific algorithms of beam search and copy mechanism are as follows:

$$w_s^{\beta} = k\text{argmax}_{w_s,\beta} P\left(w_s \mid r_{1:n}, w_1^{\beta}, \ldots, w_{s-1}^{\beta}\right) \cdot \prod_{i=1}^{s-1} P\left(w_i^{\beta} \mid r_{1:n}, w_1^{\beta}, \ldots, w_{i-1}^{\beta}\right) \tag{17}$$

$$\widehat{w}_s = \text{argmax}\,(z_s) \text{ if } w_s =< unk > \tag{18}$$

where w_s^{β} indicates the words in beam β at timestep s, and $k$argmax is the argmax function extended to *topk* scale. When inferencing text, these k words with the highest probability are inputs to the next timestep in turn, until the length of the text exceeds the pre-defined max_length. z_s is the context vector given above. Replace w_s with $\widehat{w}_s$ when the model generates <*unk*> characters at a certain timestep.

4. Experiments and analysis

We firstly introduce the prerequisite in this section (i.e., datasets, evaluation metrics, and experiment setups) and then compare SAN-T2T with several baselines and SOTAs. At last, we assess the case studies and attention visualization to reveal SAN-T2T's performance.

4.1. Datasets and evaluation metrics

In this section, we introduce the datasets we used and the evaluation metrics of the experiment.

4.1.1. Datasets

We used WikiBio (Lebret *et al.*, 2016) as the benchmark dataset, and RotoWire (Wiseman *et al.*, 2017) and WeatherGov (Liang, Jordan, and Klein, 2009) as supplementary datasets to verify the

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 1. The Wikipedia infobox of Frederick Parker–Rhodes

	Frederick Parker–Rhodes
Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Nationality	British
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
Fields	Mycology, plant pathology, mathematics, linguistics, computer science
Author abbrev. (botany)	Park–Rhodes

The introduction of his biography reads: "Arthur Frederick Parker–Rhodes (21 November 1914–2 March 1987) was an English linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist."

generalization of SAN-T2T. WikiBio contains 728,321 samples from Wikipedia, using the first paragraph of each article as the description of the corresponding infobox, as shown in Table 1. RotoWire contains 4853 samples, which are NBA basketball game summaries aligned with their corresponding box- and line-scores. WeatherGov contains 29,528 samples, each of which has 36 records of fixed length and is paired with a description text. We split WeatherGov dataset according to the proportion of training (80%), development (10%), and test (10%).

4.1.2. Evaluation metrics

BLEU, an auxiliary tool for bilingual translation quality evaluation, is often used to evaluate the quality of text generation models. The core idea is to determine the similarity between two sentences and then evaluate precision and fluency of the candidate texts through different n-grams. Since BLEU only calculates the precision of the candidate texts without considering recall, ROUGE is proposed to solve this problem, and it evaluates the quality of the generated texts through paying more attention to different n-grams' recall.

We assessed the model on table-to-text generation with BLEU-4 (Papineni *et al.*, 2002) and ROUGE-4 (Lin, 2004), and the standard scripts provided by the NLTK-3.5 and ROUGE-1.5.5 are used to calculate BLEU and ROUGE scores.

For RotoWire, we also assessed the model with the metrics from Wiseman et al. Let $\widehat{y}$ represents the reference summary and y the model output. Relation generation (RG) evaluates the precision and count of relations extracted from y that also appear in records r. CS evaluates the precision and recall of relations extracted from y that are also extracted from $\widehat{y}$. Content ordering evaluates the normalized Damerau–Levenshtein distance between the sequences of relations extracted from y and $\widehat{y}$.

4.2. Experiment setup

For WikiBio,[a] we select the most frequent 20,000 words in the training set to build the vocabulary, and the others will be encoded with <*unk*>. In the encoder, the value embedding and its corresponding field and position embedding are used as input. The largest position number is limited to 30, that is, fields with lengths longer than 30 will be truncated. <*sos*> and <*eos*> are added at the start and end of the target text, respectively. Records exceeding the pre-defined length 100 will be

[a] The codes are available at *https://github.com/ding-haijie/SAN_D2T*.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 2. Results on WikiBio (test set)

Model	BLEU	ROUGE
KN	2.21	0.38
Template KN	19.8	10.7
NLM	4.17 ± 0.54	1.48 ± 0.23
Table NLM	34.7 ± 0.36	25.8 ± 0.36
Table2Seq-Single	40.26	/
Structure-aware Seq2Seq	44.89	41.21
Two-level model	**45.77**	**41.28**
Transformer	42.80	40.30
Seq2Seq	35.18	30.48
+ position encoding	35.64	31.35
+ content selector (SAN-T2T)	39.20	34.77
+ copy mechanism	44.08	39.59
+ beam search ($k = 5$)	41.42	41.10

The boldface values show the highest values in the corresponding column.

truncated and insufficient ones will be padded with <*pad*>. We also tried different beam widths to verify the influence of beam search on the model.

Set the dimension of field embedding to 50, word embedding to 400, and position embedding to 5. Hidden layer size is 500, and batch size is 32. We also restrict the generated sentence by the max_length of 60 to avoid redundant or time-consuming generation. For the Transformer model, the hidden units of the multi-head component and the feed-forward layer are both 512, with 8 heads and 6 encoder/decoder layers.

For RotoWire, we use similar settings as WikiBio without implementing position encoding. Set the dimension of field embedding to 30, word embedding to 300. max_length is 700, and max_field is 770.

For WeatherGov, in the encoder, we only need to set the embedding vector of words in the table as input. The word embedding dimension is 300, and hidden layer size is 500. For cross entropy loss, $\gamma = 4$ is used to set the regularization term, and we also have experimented on the influence of different γ on the model.

Adam optimizer is used to perform gradient descent, the initial learning rate is 0.0003, and the gradient clipping value is 5.0.

4.3. Results and analysis

As shown in Table 2, SAN-T2T is compared with several previous works, including KN, Template KN (Heafield *et al.*, 2013), NLM, Table NLM (Lebret *et al.*, 2016), Table2Seq-Single (Bao *et al.*, 2019), Structure-aware Seq2Seq (Liu *et al.*, 2018), and Two-level model (Cao, Gong, and Zhang, 2019). Meanwhile, we provide a vanilla Seq2Seq model (without position encoding and content selector compared to SAN-T2T) and a Transformer model (Vaswani *et al.*, 2017) to conduct an ablation study.

According to Table 2, we observe that neural NLM always surpasses the statistical ones, even the vanilla Seq2Seq model can achieve better results than Table NLM, and content selector

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 3. Effects of text length on WikiBio (test set)

Text length (tokens)	BLEU	ROUGE
<10	20.31	40.29
10–20	**52.16**	**46.74**
20–30	49.14	41.56
30–40	35.27	32.37
>40	21.17	24.22

The boldface values show the highest values in the corresponding column.

Table 4. Effects of different beam width on WikiBio

	Beam width	BLEU	ROUGE
without copy mechanism	1	39.20	34.77
	2	38.05	35.36
	3	37.43	35.58
	4	37.11	35.89
	5	36.32	35.92
with copy mechanism	1	**44.08**	39.59
	2	43.39	40.68
	3	42.54	40.98
	4	41.91	41.07
	5	41.42	**41.10**

The boldface values show the highest values in the corresponding column.

significantly improves the baselines. The Transformer model based solely on attention mechanisms achieves competitive performance, having the ability that generates buffered positional encoding using sine or cosine functions according to the length of the fields (i.e., no need for additional training). Then, according to the case studies on WikiBio, although SAN-T2T often generates short texts accurately and fluently, there is still large room for improvement for long text generation (e.g., occasionally missing information that counts), which causes the low performance. Transformer performs better through replacing the chain-like forms in LSTM structure in SAN-T2T, no need to worry about exploding gradient when generating longer texts. To learn more information on this impact, we also did experiments on samples with different reference-text-length (we split WikiBio's test set into five subsets with different lengths, less than 10, between 10 and 20, between 20 and 30, between 30 and 40, and more than 40). As shown in Table 3, samples with text length between 10 and 30 fit the best, and samples with text length more than 40 the worst (samples with less than 10 tokens fit badly on BLEU score, for BLEU metric only calculates precision of the candidate texts without considering recall, and the high score on ROUGE verifies this).

Table 4 shows the impact of beam search. The BLEU score degrades as the beam width increases. The reason lies in the search strategy. Increasing the beam width makes the model generates texts that are disproportionately based on those early and non-greedy decisions, typically

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 5. Results on RotoWire (test set)

	RG		CS		CO	
Model	#	P%	P%	R%	DLD%	BLEU
TEMPL	**54.23**	**99.94**	26.99	**58.16**	14.92	8.46
WS-2017	23.72	74.80	29.49	36.18	15.42	14.19
NCP + CC	34.28	87.47	34.18	51.22	18.58	16.50
Hierarchical-k	21.17	89.46	**39.47**	51.64	18.90	17.50
DATA-TRANS	24.12	79.17	36.48	42.74	**22.40**	**20.16**
SAN-T2T	19.21	62.54	27.93	42.24	11.31	12.20
+ copy mechanism	22.95	64.24	29.17	44.66	14.70	13.08

Columns indicate relation generation (RG) count (#) and precision (P%), content selection (CS), precision (P%) and recall (R%), count ordering (CO) in normalized Damerau–Levenshtein distance (DLD%), and BLEU. These metrics are described in Section 4.1. The boldface values show the highest values in the corresponding column.

including words with relatively low probability followed by words with quite high conditional probability, leading to an overall higher probability sentence but lower precision. For example, when the k words in a certain timestep contain the character <*eos*>, then its subsequent posterior words would likely remain <*eos*> (i.e., the sentence length will be shortened, for <*eos*> will be ignored), making the sentence holding a quite high probability but badly poor performance. Increasing the beam width will also significantly reduce the inference speed (about three times slower when setting beam width to 5 in our experiments).

Results also show that the application of copy mechanism can significantly improve the model (the group with copy exceed those without copy about 5 points), and it does not need additional training because there is no need for the redundant copy-network like Gu *et al.* (2016) to determine whether words need to be copied at current timestep, for the copied words only come from the attention vector.

After finishing works on WikiBio, we also did experiments on RotoWire and WeatherGov to verify SAN-T2T's generalization. Table 5 compares SAN-T2T with TEMPL (template system), WS-2017 (Wiseman *et al.*, 2017), NCP + CC (Puduppully *et al.*, 2019), Hierarchical-k (Rebuffel *et al.*, 2019), and DATA-TRANS (Li *et al.*, 2019) on RotoWire. The model DATA-TRANS is the state of the art of RotoWire, which has a Transformer-based model that learns CS and summary generation jointly, and uses two data augmentation methods synthetic data generation and training data selection. The template system has the highest RG precision, for it's faithful to the infoboxes by design. SAN-T2T is worse than the comparison models, and the reason lies in its weak ability to generate long text and reserve information, which has already been discussed in WikiBio's results.

Table 6 compares SAN-T2T with KL (Konstas and Lapata, 2013a), MBW (Mei *et al.*, 2016), and Two-level model (Cao *et al.*, 2019) on WeatherGov. There is no need to apply copy mechanism because its vocabulary size is small and each sample has 36 records of fixed length, but it could also be concluded that beam search has little improvements, which is consistent with Mei *et al.* (2016).

For the experiments on WeatherGov, results in Table 7 show the effect of different γ on the model. BLEU score reaches the highest when $\gamma = 4$. After analyzing the original dataset, it can be seen that the most records number aligned by the reference text of all samples is 4, and the role of γ is to constrain the number of records that the model aligns. Therefore, the best result at $\gamma = 4$ is in line with expectations.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 6. Results on WeatherGov (test set)

Model	BLEU
KL	36.54
MBW	61.01
Two-level model	62.89
SAN-T2T	82.84
+ beam search ($k = 5$)	**82.90**

The boldface values show the highest values in the corresponding column.

Table 7. Effects of different γ on WeatherGov

γ	2	3	4	5	6	7	8	9
BLEU	82.18	82.65	**82.84**	82.13	82.22	81.39	80.84	80.78

The boldface values show the highest values in the corresponding column.

Table 8. The Wikipedia infobox of Alonzo H. Cushing

Name	Alonzo H. Cushing
birth_date	19 January 1841
birth_place	Delafield, Wisconsin, US
death_date	3 July 1863
allegiance	United States of America
branch	Union Army
commands	4th U.S. Light Artillery, Battery A
battles	American Civil War
awards	Medal of Honor

4.4. Case study

Three randomly selected samples from WikiBio, WeatherGov, and RotoWire and their descriptions are presented below. We will analyze the strengths as well as the remaining weaknesses of SAN-T2T for these cases.

Analysis of the texts generated for Alonzo Cushing in Tables 8 and 9 shows that SAN-T2T can learn more semantic information and even correct reasoning about that are not included in the input: "*u.s. military's highest decoration*" does not appear in the input, but the knowledge learned from other samples (the medal of honor = u.s. military's highest decoration) completes this information. When applying copy mechanism for the model, the copied word "*battery*" not only alleviates the problem of rare words but also makes up for some missing information: the ignored field *commands* appear as "*an artillery officer*" in the reference text, while SAN-T2T (without copying) outputs "*actions at the battle of <unk>*."

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 9. The generated descriptions for Alonzo H. Cushing

Model	Generated description
Reference	Alonzo Hersford Cushing (January 19, 1841–July 3, 1863) was an artillery officer in the union army during the American civil war
Transformer	Alonzo H. Cushing (January 19, 1841–July 3, 1863) was a union army officer during the American civil war and a recipient of the medal of honor, for his action in the American civil war in the army
Seq2Seq	Alonzo H. Cushing (January 19, 1841–July 3, 1863) was an American soldier who fought in the American civil war *and military decoration and the American civil war*
+ position encoding	Alonzo H. Cushing (January 19, 1841–July 3, 1863) was a union army officer in the American civil war and a recipient of *the* medal of honor
+ content selector (SAN-T2T)	Alonzo H. Cushing (January 19, 1841–July 3, 1863) was a union army officer in the American civil war and a recipient of the U.S. military 's highest decoration, the medal of honor, for his actions at the battle of <*unk*> in the American civil war
+ copy mechanism	Alonzo H. Cushing (January 19, 1841–July 3, 1863) was a union army officer in the American civil war and a recipient of the U.S. military 's highest decoration, the medal of honor, for his actions at the battle of battery in the American civil war
+ beam search ($k = 5$)	Alonzo H. Cushing (January 19, 1841–July 3, 1863) was a union army officer during the American civil war

Table 10. Human Evaluation on WikiBio (test set)

Sentence source	% fluent	% faithful	(% fluent+ mostly fluent)	(% faithful+ mostly faithful)	(k for fluent)	(for faithful)
Reference	**91%**	**86%**	**100%**	**93%**	/	/
Transformer	71%	68%	88%	84%	/	/
Seq2Seq	67%	54%	76%	70%	/	/
SAN-T2T (w/ copy)	69%	64%	85%	78%	0.251	0.334
+ beam search ($k = 5$)	72%	70%	90%	82%	/	/

The boldface values show the highest values in the corresponding column.

Human evaluation: For more intuitiveness, we perform human evaluation based on the sentences' fluency (natural and grammatical) and semantic faithfulness (supported by the records). We defined three levels of fluency as: fluent, mostly fluent, and not fluent, and the same for semantic faithfulness. Three annotators are asked to evaluate on 100 randomly selected samples from the generated and reference sentences. Specifically, we asked participants questions about RG, syntax, coherence, and brevity to estimate the output of the model. Inter-rater agreement follows Fleiss' kappa (Fleiss, 1971). For fluency, kappa score $k = 0.251$ (annotators assign decisions on the same 20 samples with the three categories), for semantic faithfulness, $k = 0.334$. Results are shown in Table 10. The reference texts achieve the highest fluency and faithfulness, and there is a considerable gap between the SAN-T2T outputs and the human-written sentences, mainly because SAN-T2T's weakness to generate long texts, often outputting repetitive sentences, and sometimes learning what should be wrong from other samples. Beam search has the ability to improve SAN-T2T's performance on all of the four measures; the top-k strategy alleviates the disadvantage that greedy search cannot get the global optimum (although with the side effect of requiring much more inference time). SAN-T2T performs almost as well as Transformer. It is also structurally

Table 11. Weather data for Northfield, Minnesota on 2009-08-1

id-0	{“type”: “temperature”, “label”: “Monday”, “time”: “6-21”, “min”: “24”, “mean”: “35”, “max”: “42”, “mode_bucket_0_20_2”: ”}
id-2	{“type”: “windSpeed”, “label”: “Monday”, “time”: “6-21”, “min”: “14”, “mean”: “20”, “max”: “24”, “mode”: “10-20”, “mode_bucket_0_20_2”: “10-20”}
id-3	{“type”: “windDir”, “label”: “Monday”, “time”: “6-21”, “mode”: “SE”, “mode_bucket_0_20_2”: ”}
id-4	{“type”: “gust”, “label”: “Monday”, “time”: “6-21”, “min”: “0”, “mean”: “24”, “max”: “32”, “mode_bucket_0_20_2”: ”}
id-10	{“type”: “precipPotential ”, “label”: “Monday”, “time”: “6-21”, “min”: “68”, “mean”: “76”, “max”: “90”, “mode_bucket_0_20_2”: ”}
id-16	{“type”: “rainChance”, “label”: “Monday”, “time”: “6-21”, “mode”: “Lkly”, “mode_bucket_0_20_2”: ”}
id-20	{“type”: “rainChance”, “label”: “Monday”, “time”: “13-21”, “mode”: “Def”, “mode_bucket_0_20_2”: ”}
id-26	{“type”: “freezingRainChance”, “label”: “Monday”, “time”: “6-21”, “mode”: “–”, “mode_bucket_0_20_2”: ”}
id-27	{“type”: “freezingRainChance”, “label”: “Monday”, “time”: “6-9”, “mode”: “Lkly”, “mode_bucket_0_20_2”: ”}

Table 12. The generated descriptions for Northfield, Minnesota on 2009-08-1

Model	Generated description
Reference	Freezing rain likely before 9 am, then rain or freezing rain likely between 9 am and noon, then rain after noon. High near 42. Breezy, with a southeast wind between 14 and 24 mph, with gusts as high as 32 mph. Chance of precipitation is 90%. Little or no ice accumulation expected
SAN-T2T ($k = 1$)	Freezing rain likely before 9 am, then rain or freezing rain likely between 9 am and noon, then rain after noon. High near 43. Breezy, with a southeast wind between 15 and 25 mph, with gusts as high as 34 mph. Chance of precipitation is 90%. Little or no ice accumulation expected

lighter than the previous best model, Transformer, with fewer parameters and faster computing speed. SAN-T2T with Beam Search performs better than Transformer, and the use of Beam Search increases the variety generated by SAN-T2T. Compared to greedy strategy sampling, Beam Search prevents SAN-T2T from failing at one step and then failing at all, resulting in better performance than Transformer.

Tables 11 and 12 are texts generated from the weather data of Northfield on 2009-02-08-1 in WeatherGov.[b] It can be seen that the text output by SAN-T2T is already very consistent with the reference, but there are still some factual errors (mainly numerical): the approximate range of temperature and wind speed has been increased (but still very close). We believe that this is because the semantic discrimination between different words encoded with one-hot is diminished, so the model’s ability to recognize different numerical tokens weaken. However, due to the large number of numerical inputs in WeatherGov, it is necessary to encode with one-hot rather than word embedding. We encode the numerical tokens to binary-array representation, which is proposed by Mei et al.

Tables 13 and 14 are the source records and generated texts from the game between Clippers and Bucks in RotoWire. SAN-T2T performs not well, missing some important information of the source records compared with the reference text. Erroneous includes the city of Bucks (which

[b] For short presentation, only necessary part of the records are shown in Table 11.

Table 13. The box score and line score for Clippers - Bucks in 2014-21-20

NAME	POS	MIN	PTS	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM	FTA	FT_PCT	OREB	DREB	REB	AST	TO	STL	BLK	PF
Matt Barnes	F	26	0	0	3	0	0	3	0	0	0	0	1	4	5	4	1	0	0	0
Blake Griffin	F	34	24	10	17	59	0	0	0	4	5	80	4	2	6	8	4	1	0	3
DeAndre Jordan	C	34	9	4	8	50	0	0	0	1	4	25	5	11	16	0	1	1	2	4
JJ Redick	G	34	23	9	15	60	5	8	63	0	0	0	0	3	3	2	1	1	0	2
Chris Paul	G	36	27	6	16	38	4	6	67	11	12	92	1	2	3	9	2	2	1	3
Glen Davis	N/A	13	2	1	2	50	0	0	0	0	0	0	0	4	4	1	0	3	0	0
Jamal Crawford	N/A	29	17	5	16	31	3	8	38	4	6	67	0	2	2	2	1	2	1	2
Hedo Turkoglu	N/A	6	0	0	0	0	0	0	0	0	0	0	0	2	2	0	1	0	0	1
Reggie Bullock	N/A	14	2	1	1	100	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Jordan Farmar	N/A	12	2	1	3	33	0	1	0	0	0	0	0	0	0	2	1	0	0	3
Jared Cunningha	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Chris Douglas-R	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Ekpe Udoh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Giannis Antetok	F	38	18	8	12	67	0	1	0	2	3	67	1	8	9	6	3	2	0	3
Johnny O'Bryant	F	6	4	2	3	67	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Larry Sanders	C	26	10	5	6	83	0	0	0	0	2	0	2	5	7	3	0	1	1	5
O.J. Mayo	G	23	3	1	6	17	0	2	0	1	3	33	0	1	1	3	2	0	0	4
Brandon Knight	G	27	8	3	10	30	2	6	33	0	0	0	1	4	5	5	4	0	0	3
Jared Dudley	N/A	30	16	7	12	58	2	4	50	0	0	0	2	6	8	3	3	2	0	2
Zaza Pachulia	N/A	20	5	1	3	33	0	0	0	3	4	75	2	5	7	2	2	0	0	1

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Table 13. Continued.

NAME	POS	MIN	PTS	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM	FTA	FT_PCT	OREB	DREB	REB	AST	TO	STL	BLK	PF
Jerryd Bayless	N/A	28	16	7	13	54	2	3	67	0	0	0	1	3	4	2	1	0	0	4
Khris Middleton	N/A	24	12	5	10	50	1	5	20	1	1	100	1	3	4	2	0	1	0	2
Kendall Marshal	N/A	18	10	4	6	67	1	3	33	1	2	50	0	1	1	3	3	0	0	0
Damien Inglis	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Jabari Parker	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Nate Wolters	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

TEAM-NAME	CITY	P_QTR1	P_QTR2	P_QTR3	P_QTR4	PTS	FG_PCT	FG3_PCT	FT_PCT	REB	AST	TO	WINS	LOSSES
Clippers	Los Angeles	28	22	32	24	106	46	46	74	41	29	12	19	8
Bucks	Milwaukee	24	28	31	19	102	53	33	53	46	29	18	14	14

The definition of table headers could be found at *https://github.com/harvardnlp/boxscore-data*.

Table 14. The generated descriptions for Clippers - Bucks

Model	Generated description
Reference	The Los Angeles Clippers (19-8) defeated the Milwaukee Bucks (14-14) 106-102 on Saturday. Los Angeles has won three of their last four games. Chris Paul paced the team with a game-high 27 points and nine assists. DeAndre Jordan continued his impressive work on the boards, pulling down 16 rebounds, and Blake Griffin and J.J. Redick joined Paul in scoring over 20 points. The Clippers have a tough stretch of their schedule coming up with the Spurs, Hawks, Warriors, and Raptors all on this week's docket. Even with the loss, Milwaukee finished their four-game Western Conference road trip 2-2, a job well done by the developing squad. In the three games since Jabari Parker went down with a season-ending ACL injury, coach Jason Kidd has cut the umbilical cord they had on Giannis Antetokounmpo. He played over 37 min for the second straight game Saturday, which is ten more minutes than his season average of 27 min per game. Larry Sanders returned to the starting lineup after sitting out Thursday's game on a league mandated one-game suspension. Ersan Ilyasova (concussion) and John Henson (foot) remain out, and it seems Ilyasova may be closer to returning than Henson
DATA-TRANS	The Los Angeles Clippers (19-8) defeated the Milwaukee Bucks (14-14) 106-102 on Saturday. Los Angeles stopped their two-game losing streak with the win. Jamal Crawford paced the team with a game-high 17 points in 29 min off the bench. Crawford shot 9 -of-16 from the field and 3-of-8 from downtown. He had nine assists, two rebounds, and two steals in 29 min. Blake Griffin had 24 points, eight assists, six rebounds, and one steal in 34 min. The Clippers will go on the road to face the Denver Nuggets on Monday. Milwaukee has lost two straight and is now 9-2 in their last 10 games. Jabari Parker (ankle) didn't play Saturday as he recorded a double-double with 18 points and nine rebounds. Giannis Antetokounmpo (8-12 FG, 2 -1 3Pt, 2-3 FT) and nine rebounds in 38 min off the bench. The Clippers will stay home and host the Brooklyn Nets on Monday
SAN-T2T (w/ copy)	The host Los Angeles Clippers took down the visiting Los Angeles Bucks on Saturday, 106-102. Clippers (19-8) is now on a three-game winning streak. The Bucks (14-14) saw tough play from its stars, who had been a rocky part of the season, the team had a tough shooting from the field, hitting 46% of their shots from beyond the arc. Crawford shot over 31% from the field and hit 3-of-8 from three-point range. The Clippers saw strong play from its bench, as the team shot over 46% from the field and hit 29 assistance. Chris Paul scored over 27 three points. The Bucks (14-14) have now lost three straight games. Damien Inglis didn't play, and they 'll look to keep rolling on Monday against the Clippers

should be Milwaukee), the percentage of the three-pointers made by Bucks (which should be 33), and wrong inferences (the Bucks lost three straight games, and need to fight against Clippers on Monday). As the text gets longer, the generation quality degrades, that is, generating repetitive sentences, which further impairs the model's performance.

4.5. Attention visualization

Figure 5 shows an example of a heat-map of the attention vector based on the text generated for John Uzzell Edwards. The sub-figure above comes from the vanilla Seq2Seq model, while the below from SAN-T2T. Both have applied the copy mechanism. The reference text of this sample is "*john uzzell edwards* (10 October 1934–5 March 2014) was a welsh painter," which consists of the SAN-T2T model. Decisive information is: {nationality: welsh}, {field: painting}, {movement: pure painting}, {awards: granada arts fellowship}. It can be seen that Seq2Seq model generates some duplicate contents (i.e., "*pure and pure*" aligns to the *movement* field) and has lost some critical information (e.g., "*welsh painter*" has been missed, while generating the sentence "*pure artist*"). However, SAN-T2T generates descriptions that are consistent with the reference text. The fields *name, birth_date, death_date, occupation* and other information are all obtained from the correct fields. For example, "*john uzzell edwards*" and "*welsh Painter*" are from the *name* field and "*welsh pure painting*," respectively. This suggests that the enhanced attention mechanism can model the relationship between records through the content selector and can accurately map the alignment between the generated text and the fields.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

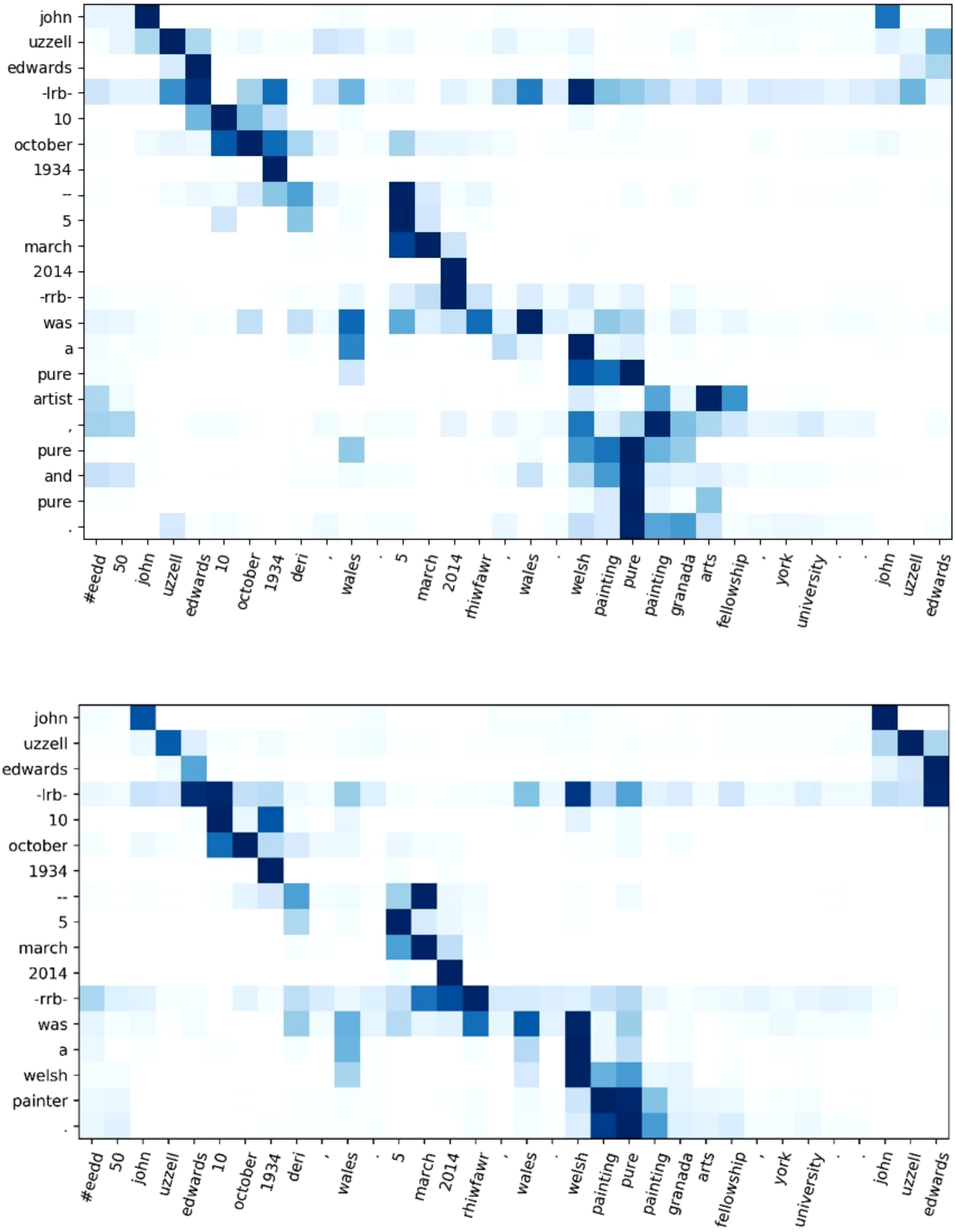


Figure 5. An example of visualization for the attention. The image above is the attention from Seq2Seq model, while the below one is from SAN-T2T. The vertical axis represents the text generated by the model, while the horizontal represents the fields' value. -lrb- and -rrb- indicate brackets (). Deeper colors depict a higher attention score.

5. Conclusion and future work

We propose a novel Seq2Seq model that further enhances the attention mechanism and conduct experiments on table-to-text generation. The model SAN-T2T is mainly composed of a field-content selective encoder and a decoder with the attention mechanism. To utilize structure in the tables, an additional gate is added to the LSTM cell in the encoder to integrate position information. Besides, the application of beam search and copy mechanism further improves the performance of SAN-T2T. Attention visualization and comparison with baselines show that SAN-T2T is far superior to these models.

Work of this research has some practical implications. First, the fact that neural language models always have better performance than statistical methods implies that the automated table-to-text generator can be designed by neural networks rather than hand-built feature engineering with statistical approaches. Second, using multi-layered features rather than simplex fields gets better results in table-to-text tasks. Third, it is possible to apply the selective attention network or its variant to other NLP tasks such as information extraction and text generation. We will research for this possibility in the future and focus on how to improve the ability of the neural language models to correctly understand the records in the tables and learn the other latent information (e.g., multi-level location. We only apply local addressing in SAN-T2T, which represents inner-record information. However, inter-record relevance within the tables is also significant, and it would be challenging to explore these information sources). SAN-T2T's weakness to generate long texts affects its performance; we will also explore the possibility of solving this length issue in the future (adapting Transformer-based models to solve the gradient-exploding problem will be a possible direction).

Acknowledgments. We would like to thank the reviewers for their comments in advance to help us improve the quality of this paper.

Financial support. This work was supported by the National Natural Science Foundation of China under Grant 62072255.

Declaration of competing interest. The authors declare none.

References

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

Bao J., Tang D., Duan N., Yan Z., Zhou M. and Zhao T. (2019). Text generation from tables. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **27**(2), 311–320.

Barzilay R. and Lapata M. (2005). *Collective content selection for concept-to-text generation. Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, Vancouver, British Columbia, Canada: ACL, pp. 331–338.

Cao J., Gong J. and Zhang P. (2019). *Two-level model for table-to-text generation. Proceedings of the 2019 International Symposium on Signal Processing Systems*, Beijing, China: ACM, pp. 121–124.

Chen K., Li F., Hu B., Peng W., Chen Q., Lv Y. and Yu H. (2020a). Neural data-to-text generation with dynamic content planning. *Knowledge-Based Systems*, **215**, 106610.

Chen W., Chen J., Su Y., Chen Z. and Wang W. Y. (2020b). *Logical natural language generation from open-domain tables. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Seattle, Washington, USA: ACL, pp. 7929–7942.

Chen W., Su Y., Yan X. and Wang W. Y. (2020c). *KGPT: Knowledge-grounded pre-training for data-to-text generation. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA: ACL, pp. 8635–8648.

Chen Z., Chen W., Zha H., Zhou X., Zhang Y., Sundaresan S. and Wang W. Y. (2020d). *Logic2Text: High-fidelity natural language generation from logical forms, Findings of the Association for Computational Linguistics: EMNLP 2020*, Seattle, Washington, USA: ACL, pp. 2096–2111.

Fleiss J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin* **76**(5), 378–382.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Gong H., Bi W., Feng X., Qin B., Liu X. and Liu T. (2020). *Enhancing content planning for table-to-text generation with data understanding and verification. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, ACL, pp. 2905–2914.

Gong H., Sun Y., Feng X. C., Qin B., Bi W., Liu X. J. and Liu T. (2020). *TableGPT: Few-shot table-to-text generation with table structure reconstruction and content matching. Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain: International Committee on Computational Linguistics, pp. 1978–1988.

Graves A., Mohamed A. R. and Hinton G. (2013). *Speech recognition with deep recurrent neural networks. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada: IEEE, pp. 6645–6649.

Gu J., Lu Z., Li H. and Li V. O. (2016). *Incorporating copying mechanism in sequence-to-sequence learning. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany: ACL, pp. 1631–1640.

Heafield K., Pouzyrevsky I., Clark J. H. and Koehn P. (2013). *Scalable modified Kneser-Ney language model estimation. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria: ACL, pp. 690–696.

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

Jean S., Cho K., Memisevic R. and Bengio Y. (2015). *On using very large target vocabulary for neural machine translation. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China: ACL, pp. 1–10.

Jiang N., Chen J., Zhou R. G., Wu C., Chen H., Zheng J. and Wan T. (2020). PAN: Pipeline assisted neural networks model for data-to-text generation in social internet of things. *Information Sciences* **530**, 167–179.

Kondadadi R., Howald B. and Schilder F. (2013). *A statistical NLG framework for aggregated planning and realization. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria: ACL, pp. 1406–1415.

Konstas I. and Lapata M. (2013a). *Inducing document plans for concept-to-text generation. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington: ACL, pp. 1503–1514.

Konstas I. and Lapata M. (2013b). A global model for concept-to-text generation. *Journal of Artificial Intelligence Research* **48**, 305–346.

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

LeCun Y., Bengio Y. and Hinton G. (2015). Deep learning. *Nature* **521**(7553), 436–444.

Li G., Crego J. and Senellart J. (2019). *Enhanced transformer model for data-to-text generation. Proceedings of the 3rd Workshop on Neural Generation and Translation*, Hong Kong, China: ACL, pp. 148–156.

Liang P., Jordan M. I. and Klein D. (2009). *Learning semantic correspondences with less supervision. 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*, Suntec, Singapore: ACL, pp. 91–99.

Lin C. Y. (2004). ROUGE: A package for automatic evaluation of summaries, *Text Summarization Branches Out*. Barcelona, Spain: ACL, pp. 74–81.

Liu M., Mu Z., Sun J. and Wang C. (2020). *Data-to-text generation with pointer-generator networks. 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, IEEE, pp. 244–251.

Liu T., Wang K., Sha L., Chang B. and Sui Z. (2018). *Table-to-text generation by structure-aware Seq2seq learning. Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana: The Association for the Advancement of Artificial Intelligence (AAAI), pp. 4881–4888.

Luong M. T., Sutskever I., V.Le Q., Vinyals O. and Zaremba W. (2015, *Addressing the rare word problem in neural machine translation, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Beijing, China: ACL, pp. 11–19.

Ma S., Yang P., Liu T., Li P., Zhou J. and Sun X. (2019). *Key fact as pivot: A two-stage model for low resource table-to-text generation. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: ACL, pp. 2047–2057.

Mei H., Bansal M. and Walter M. R. (2016). *What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California: ACL, pp. 720–730.

Oya T., Mehdad Y., Carenini G. and Ng R. (2014). *A template-based abstractive meeting summarization: Leveraging summary and source text relationships. Proceedings of the 8th International Natural Language Generation Conference*, Philadelphia, USA: ACL, pp. 45–53.

Papineni K., Roukos S., Ward T. and Zhu W. J. (2002). *BLEU: A method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania: ACL, pp. 311–318.

https://doi.org/10.1017/S135132492300013X Published online by Cambridge University Press

Parikh A., Wang X., Gehrmann S., Faruqui M., Dhingra B., Yang D. and Das D. (2020). *ToTTo: A controlled table-to-text generation dataset. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, ACL, pp. 1173–1186.

Puduppully R., Dong L. and Lapata M. (2019). *Data-to-text generation with content selection and planning. Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii: AAAI, pp. 6908–6915.

Puduppully R. and Lapata M. (2021). Data-to-text generation with macro planning. *Transactions of the Association for Computational Linguistics* **9**, 510–527.

Qin G., Yao J. G., Wang X., Wang J. and Lin C. Y. (2018). *Learning latent semantic annotations for grounding natural language to structured data. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: ACL, pp. 3761–3771.

Rebuffel C., Soulier L., Scoutheeten G. and Gallinari P. (2019). A hierarchical model for data-to-text generation. *arXiv preprint*. 1912.10011.

Rebuffel C., Roberti M., Soulier L., Scoutheeten G., Cancelliere R. and Gallinari P. (2022). Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery* **36**(1), 318–354.

Reiter E. (2007). *An architecture for data-to-text systems. Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*, Saarbrücken, Germany: DFKI GmbH, pp. 97–104.

Reiter E. and Dale R. (2000). *Building Natural Language Generation Systems*. Cambridge: Cambridge University Press.

See A., Liu P. J. and Manning C. D. (2017). *Get to the point: Summarization with pointer-generator networks. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada: ACL, pp. 1073–1083.

Sha L., Mou L., Liu T., Poupart P., Li S., Chang B. and Sui Z. (2018). *Order-planning neural text generation from structured data. Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana: AAAI, pp. 5414–5421.

Sutskever I., Vinyals O. and Le Q. V. (2014). *Sequence to sequence learning with neural networks. Proceedings of the Advances in Neural Information Processing Systems*, Montréal, Canada: Conference on Neural Information Processing Systems (NIPS), pp. 3104–3112.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł. and Polosukhin I. (2017), Attention is all you need, *arXiv preprint*. 1706.03762.

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

Wiseman S., Shieber S. M. and Rush A. M. (2018). *Learning neural templates for text generation. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: ACL, pp. 3174–3187.

Zhang B., Xiong D. and Su J. (2020). Neural machine translation with deep attention. *IEEE/ACM Transactions on Pattern Analysis and Machine Intelligence* **42**, 154–163.

Zhang Y., Zhang P. and Yan Y. (2019). Tailoring an interpretable neural language model. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **27**, 1164–1178.

Cite this article: Ding H and Xu X (2024). SAN-T2T: An automated table-to-text generator based on selective attention network. *Natural Language Engineering* **30**, 429–453. https://doi.org/10.1017/S135132492300013X