Doctoral Dissertation

Study of Content Order-Controllable MR-to-Text Generation

Keisuke Toyama
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Satoshi Nakamura
Augmented Human Communication Lab. (Division of Information Science)

Submitted on February 20, 2024

A Doctoral Dissertation
submitted to Graduate School of Science and Technology,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Keisuke Toyama

Thesis Committee:

Supervisor	Satoshi Nakamura
	(Professor, Division of Information Science)
	Taro Watanabe
	(Professor, Division of Information Science)
	Katsuhito Sudoh
	(Associate Professor, Division of Information Science)

Study of Content Order-Controllable MR-to-Text Generation*

Keisuke Toyama

Abstract

Content order is critical in natural language generation (NLG) for emphasizing the focus of a generated text passage. In this paper, we propose a novel MR (meaning representation)-to-text method that controls the order of the MR values in a generated text passage based on the given order constraints. We develop a refined MR-text dataset with additional value order annotations to train our order-controllable MR-to-text model. We also use it to train a text-to-MR model to check whether the generated text passage correctly reflects the original MR. Furthermore, we augment the dataset with synthetic MR-text pairs to mitigate the discrepancy in the number of non-empty attributes between the training and test conditions and use it to train another order-controllable MR-to-text model. Our proposed methods demonstrate better NLG performance than the baseline methods without order constraints in automatic and subjective evaluations. In particular, the augmented dataset effectively reduces the number of deletion, insertion, and substitution errors in the generated text passages.

Keywords:

Controllable Text Generation, Data Augmentation, Data-to-Text, Meaning Representation, Natural Language Generation

*Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, February 20, 2024.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Natural Language Generation (NLG) [16, 27] is a technology that automatically generates sentences in the language used by humans in everyday life from certain inputs. With the recent development of Artificial Intelligence (AI) technology, there is a growing demand for machines that can communicate with humans through natural language, and that is why NLG is one of the most important technologies in this field.

The main inputs to NLG are sentences and data. Applications for sentence input include, for example, machine translation that converts sentences in one language into sentences in another language [38, 64, 100], dialogue response generation that uses the history of previous dialogue as input [44], summarization that summarizes longer sentences into shorter sentences [12, 22], question and answer generation that generates examples of questions and answers from instructions for a certain product or service [17], and style transfer that converts sentences spoken by a person into sentences as if spoken by a certain character [40].

On the other hand, many applications for data input, called *data-to-text*, have been developed that input data [27]. The first commercial systems was weather forecasts from weather data [29], but there are many other systems including soccer reports [86, 9], virtual newspapers from sensor data [61] and news reports on current affairs [53], text addressing environmental concerns [79, 95, 91], summaries of patient information in clinical contexts [37, 33, 68, 26, 5], interactive information about cultural artefacts [65, 81], text intended to persuade [6], text intended to motivate behaviour modification [74], the generation of weather summary text from sensor-observed information such as temperature, weather, wind speed, etc., at a specific location [4], the generation of tourist information from sensor information such as congestion and bus operation information at a tourist spot [46] (Figure 1), the generation of summaries of sports matches [97]. Captioning image [80] and sound data [60] is another example of NLG.

Since Kukich [48] proposed a data-to-text method for generating stock reports from a daily stock quote database, many data-to-text methods have been studied. The first approach for data-to-text was *template-based* methods [73, 58, 99, 96]

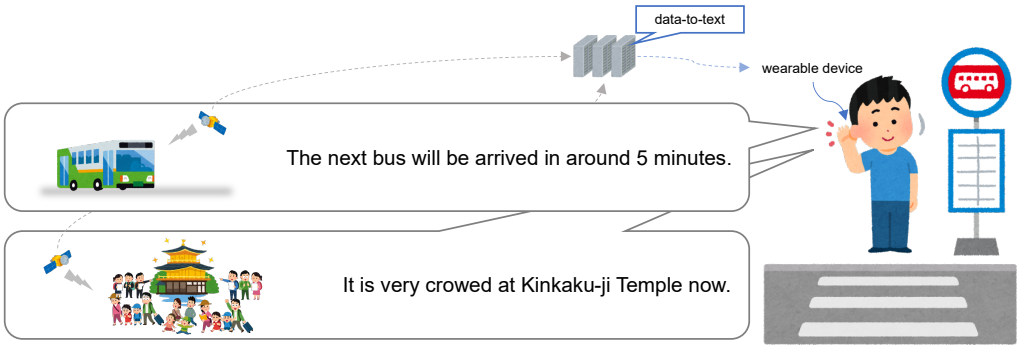

Figure 1. Example of systems with data-to-text

(described in Section 2.1.1). The disadvantage of templates is that they are labor-intensive if constructed by hand. In addition, all templates for numerous linguistic variations must be prepared in advance.

Statistical approaches have been developed to tackle the issue. The most well-known approaches are *language model-based* methods (described in Section 2.1.2) that obtain statistical knowledge from large corpora. For example, one of the earliest approaches of Langkilde-Geary et al. [49] relies on statistical knowledge in the form of N-grams. However, their system was still hand-crafted. Recent advances in the computing environment have led to a growing body of research on deep neural networks (DNNs), which handle large amounts of data. In particular, several data-to-text studies have been published since Sutskever et al. [83] provided the potential of DNN.

Among the various formats of input data in data-to-text, such as concepts, tables, knowledge graphs and RDF, this study deals with meaning representations (MR) that consist of a set of pairs of a short text passage and corresponding MR with some attribute-value pairs as shown in Table 1, because the structure of MR is simple and the most tractable format.

Here, let us now consider the relationship between data and text. Not necessarily only one sentence corresponds to certain data, but it can correspond to various sentences with similar meanings. For example, if the input data for the report of a football game is {`player`: "James", `score for team`: "2nd", `time`: "45 minutes"}, one possible corresponding sentence is "*James scored his team's second goal in the 45th minute.*" "*The second goal for James' team was scored by*

him in the 45th minute." and "*In the 45th minute, James found the back of the net for his team's second goal.*" can be some other examples. Those sentences do not have exactly the same meaning but have different nuances and, therefore, need to be used according to the situation and context in which they are used. Therefore, in data-to-text, controlling how sentences are generated is essential.

This paper examines an MR-to-text method that controls the order of contents in which data appears in sentences since the order is crucial for data-to-text that emphasizes important words or phrases in the generated sentence. Here, we explain two examples of the importance of controlling the content order.

Table 1. Example of E2E dataset

MR	Attribute	Value
	`name`	Wildwood
	`eatType`	restaurant
	`food`	Italian
	`priceRange`	(empty)
	`customer rating`	(empty)
	`area`	riverside
	`familyFriendly`	no
	`near`	Raja Indian Cuisine
Text	Wildwood is a restaurant that serves Italian food located near Raja Indian Cuisine in the area of riverside. Unfortunately, it is not kid friendly.	

(1) Preserving the order to avoid changing the focus of a text passage: Suppose we replace the value "*Italian*" of the `food` with "*French*" in Table 1. In this case, the ideal text passage becomes "*Wildwood is a restaurant that serves French food located near Raja Indian Cuisine in the area of riverside. Unfortunately, it is not kid friendly.*" In this passage, the word "*Italian*" is changed to "*French*," although the content order is unchanged. In the reference text passage and the ideal text passage, the "*no*" of the `familyFriendly` attribute is emphasized by placing an independent sentence at the end of the text passage: "*Unfortunately, it is not kid friendly.*" However, if we cannot control the order of a data-to-text system, it may generate a text passage with an unintended order: "*Wildwood is a restaurant that is not family friendly. It serves French food located near Raja Indian Cuisine in the riverside.*" This text passage does not

emphasize the "*no*" of the `familyFriendly` attribute because of the location of the phrase "*that is not family friendly*" in the middle of the sentence.

(2) Controlling the order to change the focus of a text passage: Suppose we emphasize the value "*riverside*" of the `area` attribute and "*Raja Indian Cuisine*" of the `near` attribute in Table 1. In this case, these values should appear at the beginning of the passage: "*In the riverside area near Raja Indian Cuisine, you can visit a restaurant called Wildwood where Italian food is served. It is not family-friendly.*"

Although some previous research on MR-to-text with order control exists [52, 82, 57, 55, 98, 76], no research can control order with high precision and generate high-quality sentences due to problems such as poor sentence generation performance, poor order control performance, or lack of evaluation experiments.

1.2 Contributions of This Dissertation

In this study, we propose a new method for converting MR to text that can manage the sequence of MR values in the generated text passage based on the provided order constraints. We developed an improved MR-text dataset with extra order annotations to train our model, which can control the order of MR-to-text. This dataset also trains a model that converts text to MR to verify if the text passage generated accurately represents the original MR. Additionally, we augment the dataset with artificial MR-text pairs to lessen the difference in the count of non-empty attributes between the training and testing conditions and use it to train another model that can control the order of MR-to-text. Our proposed methods show superior NLG performance compared to the baseline methods without order restrictions in automatic and subjective evaluations. Notably, the augmented dataset effectively decreases the number of deletion, insertion, and substitution errors in the generated text passages.

1.3 Outline

The structure of the remaining chapters of this dissertation is as follows. Chapter 2 looks at what data-to-text technologies have been developed so far. Various efforts to address the content order-controllability issue are also introduced, which

is the subject of this thesis. Furthermore, some metrics for automatic evaluation are introduced. Chapter 3 introduces a new dataset that we developed and used for this study. Chapter 4 proposes a method to control content order using the dataset described in Chapter 3. Chapter 5 summarises the remaining issues and future works.

2. Controllable Data-to-Text

2.1 Data-to-Text

Data-to-text is a technology to generate natural language sentences from data input. There are two main methods for data-to-text: template-based method and language model-based method.

2.1.1 Template-based Method

As shown in Figure 2, there is a method for generating sentences by creating templates from a large corpus collected in advance and selecting the appropriate template to embed information according to the data (entity) to be conveyed. For example, Yamazaki et al. [99] proposed a method for automatic template generation and selection to generate text for a cooking recipe. In this method, it is possible to generate highly accurate sentences that are close in textual aspect to sentences in the original corpus. However, since this method selects templates by matching with entities, it is challenging to handle entity patterns that are not included in the corpus. Therefore, to handle a variety of entities, it is necessary to collect a variety of corpora in advance. In addition, since the endings and verbs may change depending on the entity's value, it is necessary to prepare various templates to accommodate them in advance. Wiseman et al. [96] proposed a neural generation system using a hidden semi-Markov model decoder, which learns latent, discrete templates jointly with learning to generate. However, the generated templates cannot express any patterns of attributes that are not included in training data. Furthermore, the performance of the method is worse than that of a method which uses a language model-based text generation model.

2.1.2 Language Model-based Method

Statistical sentence generation methods use language models [102] (Figure 3). A language model is learned from a large corpus, and sentences are generated by selecting words based on the probability of occurrence obtained from the model and linking them. For example, methods based on the N-gram model

$$P(w_t) = P(w_t|w_{(t-1-(N-1):t-1)}), \tag{1}$$

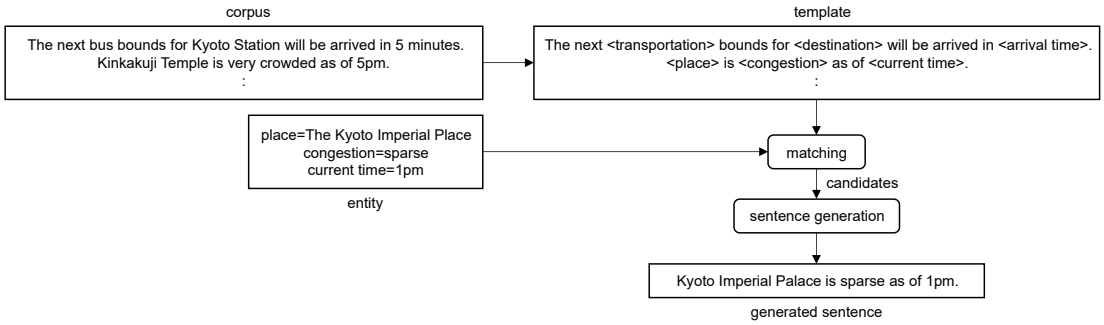

Figure 2. Example of template-based method

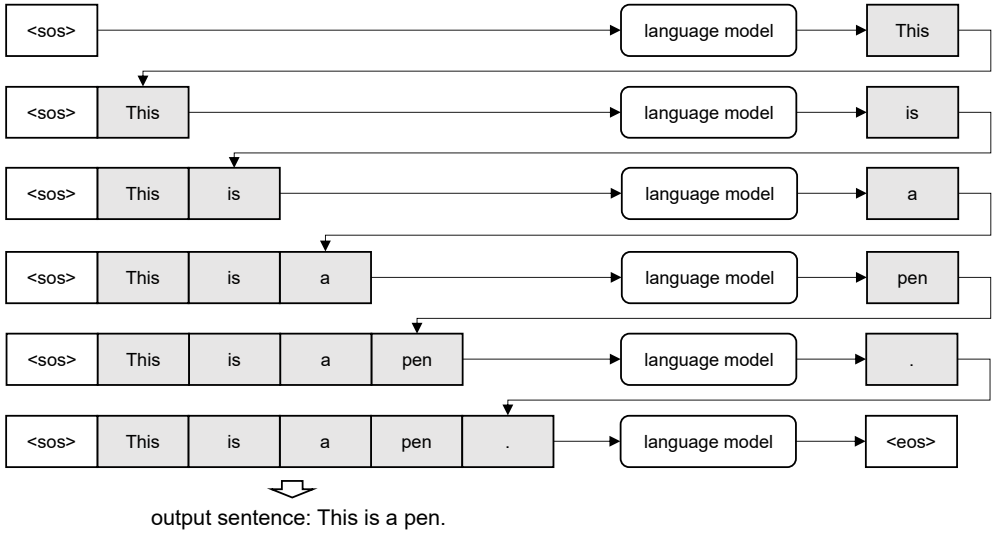

Figure 3. Example of language model-based method

in which the probability of occurrence of a word w_t depends only on the previous N words. On the other hand, the neural language model is trained as probabilistic classifiers to predict the probability distribution for all words t in all vocabulary V as:

$$P(w_t) = P(w_t|\text{context}) \forall t \in V. \tag{2}$$

Typically, context can be expressed as preceding k words:

$$P(w_t) = P(w_t|w_{t-k}, ..., w_{t-1}). \tag{3}$$

2.1.3 Sequence-to-Sequence Method

The language model is often constructed as an encoder-decoder model [11] as shown in Figure 4. The encoder converts the input sequence ($\boldsymbol{x}$) into a vector (C), and the decoder uses the vector to generate sentences ($\boldsymbol{w}$) as:

$$P(\boldsymbol{w}|\boldsymbol{x};\theta) = \prod_{t=1}^{M} p(w_t|w_1, w_2, ..., w_{t-1}, \boldsymbol{x}), \tag{4}$$

$$w_t = \text{Decoder}(w_1, w_2, ..., w_{t-1}, C), \tag{5}$$

$$C = \text{Encoder}(\boldsymbol{x}), \tag{6}$$

where θ is the parameter of the model.

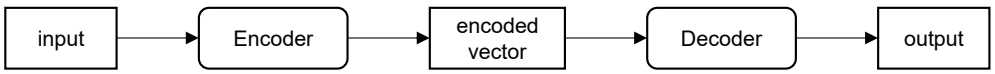

Figure 4. Encoder-Decoder Model

Now consider data-to-text, where data is the input. In this case, the data necessary to generate a sentence is converted into a vector by the encoder, and the information is converted into a sentence by the decoder. If the input data format is Meaning Representation (MR), one possible composing method of input is a sequence of attribute-value pairs as shown in Figure 5. In this case, both the input and output are sequence. So, this strategy is called *sequence-to-sequence* [84]. Nowadays, many encoder-decoder systems utilize the sequence-to-sequence strategy [19, 84].

As sequence-to-sequence models, LSTM (Long Short-Term Memory) [34] and Transformer [92] are widely used these days.

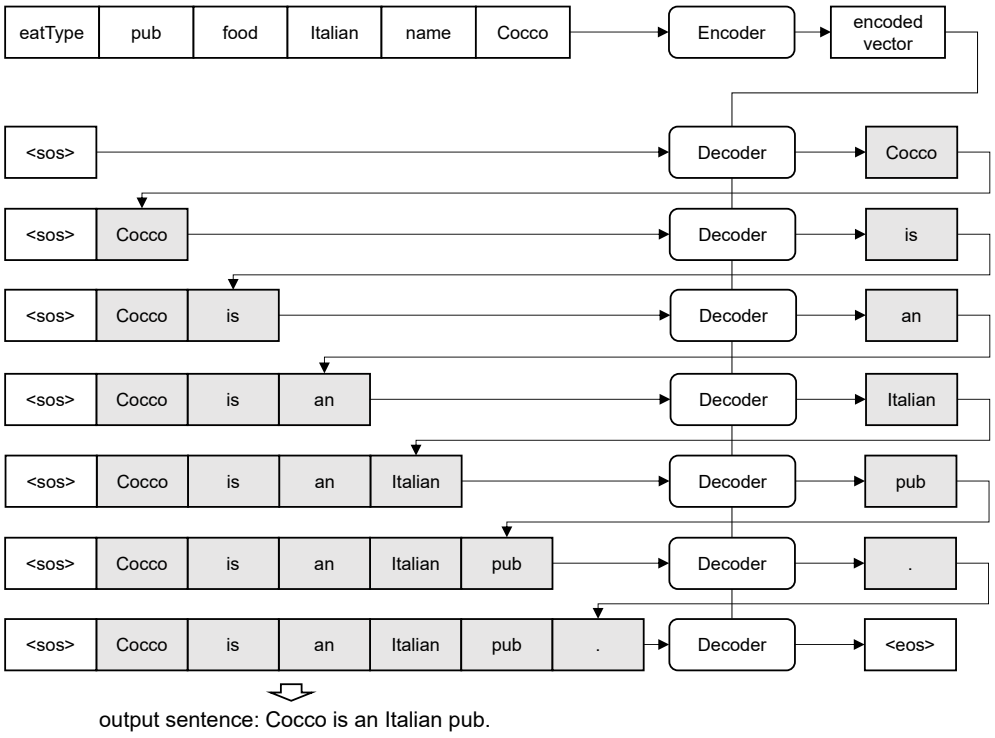

Figure 5. Example of encoder-to-decoder model for data-to-text

LSTM Recurrent neural networks (RNNs) are excellent networks for handling variable-length series. The RNN model uses the hidden state vector of the previous time (h_{t-1}) and the input vector of the current time (x_t) to update the hidden state vector of the current time (h_t), as shown in Figure 6. As shown in Figure 7, the RNN layer consists of a matrix sum-of-products operation and an activation function. As the activation function, Hyperbolic tangent (tanh) is often used.

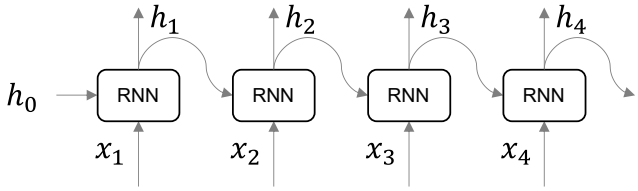

Figure 6. Example of RNN

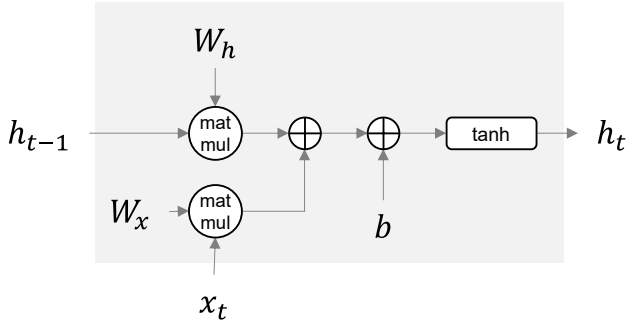

Figure 7. RNN architecture

However, RNNs have the *vanishing gradient problem* and *exploding gradient problem*, where the error gradient converges to zero or diverges to infinity as it propagates through the computational graph during backpropagation, making it impossible to train the model. To simplify the discussion, assume that the following equation is processed in the RNN module, ignoring the activation function.

$$h_t = W_x x_t + W_h h_{t-1} + b. \tag{7}$$

Here, let us consider the gradient of the error propagation from the output h_M

at time M to the output h_1 at time 1.

$$\frac{dh_M}{dx_1} = \frac{dh_M}{dh_{M-1}} \cdot \frac{dh_{M-1}}{dh_{M-2}} \cdot \cdots \cdot \frac{dh_2}{dh_1} \cdot \frac{dh_1}{dx_1} \tag{8}$$

$$= (W_h \cdot W_h \cdot \cdots \cdot W_h \cdot W_x)^T \tag{9}$$

$$= (W_h^{M-1} W_x)^T, \tag{10}$$

where T indicates transposition. Thus, it converges to 0 (vanishing gradient) or $\pm\infty$ (exploding gradient) depending on the singular value of the matrix W_h.

LSTM (Long Short-Term Memory) [34] is a type of RNN that has been proposed as a solution to the vanishing/exploding gradient problem of RNNs; LSTM is characterised by having a dedicated memory cell c_t, which stores LSTM information from the past up to time t. Here, c_t is only used inside the LSTM, and the only external output is h_t, as shown in Figure 8.

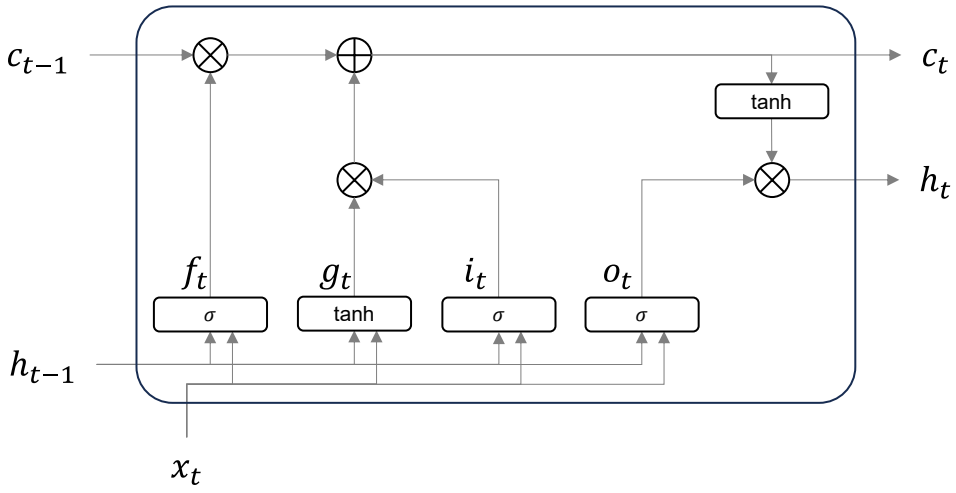

Figure 8. LSTM architecture

The LSTM includes the following three gates: *forget gate* (f_t), *input gate* (i_t), and *output gate* (o_t). The output h_t is calculated as follows. The value of the memory cell c_t is first scaled to the range $[-1, 1]$ using the tanh function ($\tanh(c_t)$). Next, the output gate (o_t) determines whether the contents of the memory cell should be output or not as a value in the range $[0, 1]$ using a sigmoid function based on the input x_t and the previous output h_{t-1} as:

$$o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o), \tag{11}$$

where σ indicates the sigmoid function. Then, the scaled value is multiplied by the value of the output gate:

$$h_t = o_t \odot \tanh(c_t), \tag{12}$$

where $\odot$ indicates the Adamar product. The memory cell values are updated as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{13}$$

where the new candidate value for the memory cell g_t is calculated from the input x_t and the previous output h_{t-1} as:

$$g_t = \tanh(W_x x_t + W_h h_{t-1} + b). \tag{14}$$

Next, it is determined whether the obtained candidate value g_t can be added to the memory cell as a value of $[0, 1]$ by a sigmoid function using the input gate as:

$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i), \tag{15}$$

then the obtained value is multiplied by g_t ($i_t \odot g_t$ in Eqn (13)) and added to the memory cell. At that time, it is also determined whether the contents of the memory cell can be forgotten, as a value in the range $[0, 1]$ by the forget gate (f_t) as:

$$f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f), \tag{16}$$

then multiplied by the value of the memory cell ($f_t \odot c_{t-1}$ in Eqn (13)).

In LSTM, error propagation is mainly considered using the memory cell equation (Eqn (13)). Consider the gradient of error propagation from the memory C_M at time M to the new memory cell g_1 at time 1.

$$\frac{dC_M}{dg_1} \simeq \frac{dC_M}{dC_{M-1}} \cdot \frac{dC_{M-1}}{dC_{M-2}} \cdot \cdots \cdot \frac{dC_2}{dC_1} \cdot \frac{dC_1}{dg_1} \tag{17}$$

$$= \mathrm{diag}(f_M \odot f_{M-1} \odot \cdots \odot f_2 \odot i_1) \tag{18}$$

As shown in this equation, gradient disappearance will not occur if learning can be done so that for each element of the forget gate, the value is 1 when propagation is required and 0 when it is not required.

Figure 9 shows an example of an encoder-decoder model using LSTM. The encoder outputs a vector of hidden layers at the last time; the decoder takes the

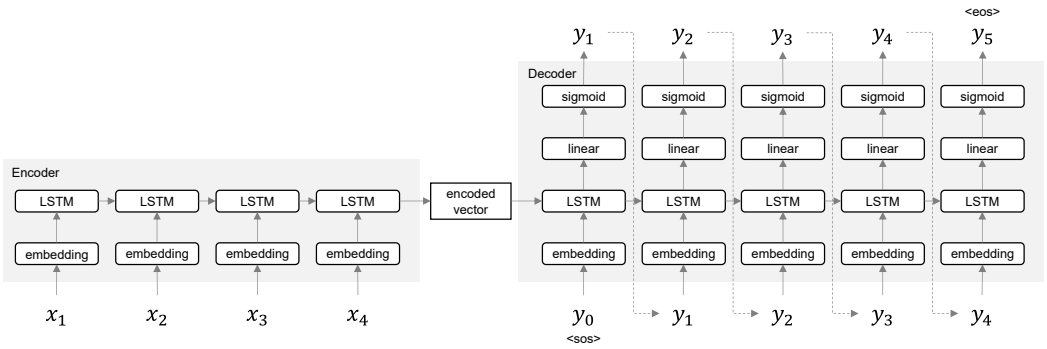

Figure 9. Encoder-Decoder model using LSTM

encoder output vector and the special symbol token (e.g., $\langle sos \rangle$) indicating the start of sentence generation as input at the first time, and the previous output is used as the input at subsequent times. Subsequently, the decoder repeats the generation process using the output tokens as input to the next LSTM layer. It terminates the process when the special symbol token (e.g., $\langle eos \rangle$) indicating the end of sentence generation is output. Here, the output hidden state vector of the LSTM encoder generated at each time only inherits information from the past and does not use information from the future. Therefore, a better balanced hidden state vector can be generated by processing in both directions, from the past to the future and from the future to the past, as shown in Figure 10. This method is called *Bi-directional LSTM*.

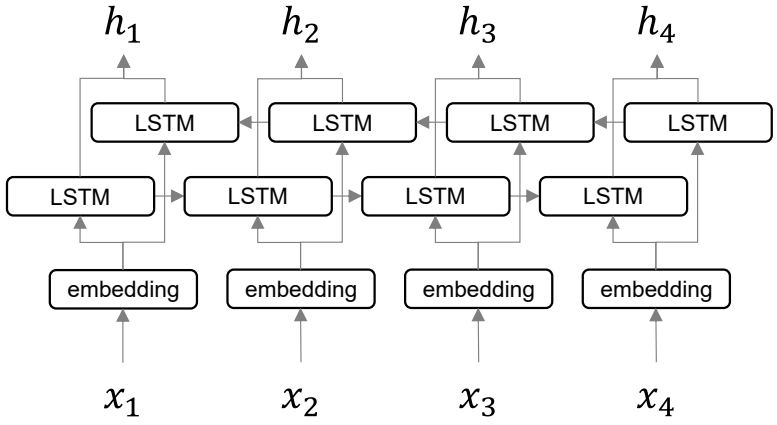

Figure 10. Bi-directional LSTM

Attention In the encoder-decoder model using LSTM described so far, the length of the encoder output vector is fixed regardless of the input data length. Thus, when the input data is long, it may not be possible to embed data features well. Therefore, changing the length of the vectors generated according to the input data length is desirable. To address this problem, *attention* mechanism [3] was developed. The output of the encoder at each time point is stored as a key vector k_i and a value vector v_i, as shown in Figure 11. Typically, these vectors are vectors for which the linear module is applied to the LSTM output. A score function obtains a score for each time using the query vector (q) based on the LSTM output vector at the decoder and the key vector at each time (k_i) in the encoder, as shown in Figure 12. The score is then multiplied by the value of the softmax process and the value vector v_i. The weighted average of these is used as the feature vector at the decoder. The most commonly used score function is scaled dot product [92]:

$$f(q,k) = \frac{q^T k}{\sqrt{d_k}}, \tag{19}$$

where d_k is the number of dimensions of the vector k.

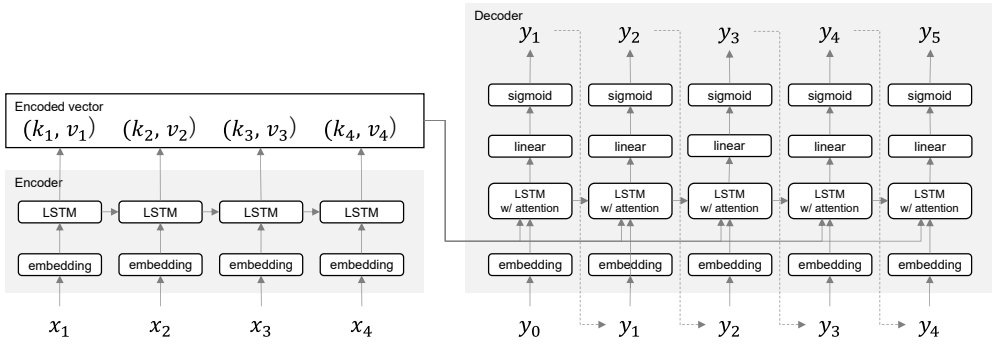

Figure 11. LSTM with attention

Transformer *Transformer* is a neural network model consisting of attention modules without convolution or recursion (Figure 13). In particular, it is characterized by using a relevance score between tokens, called *self-attention*, calculated using only the tokens contained in a single sequence.

Unlike the attention mechanism described earlier, the self-attention mechanism directs attention to its series. This mechanism is used for feature extraction

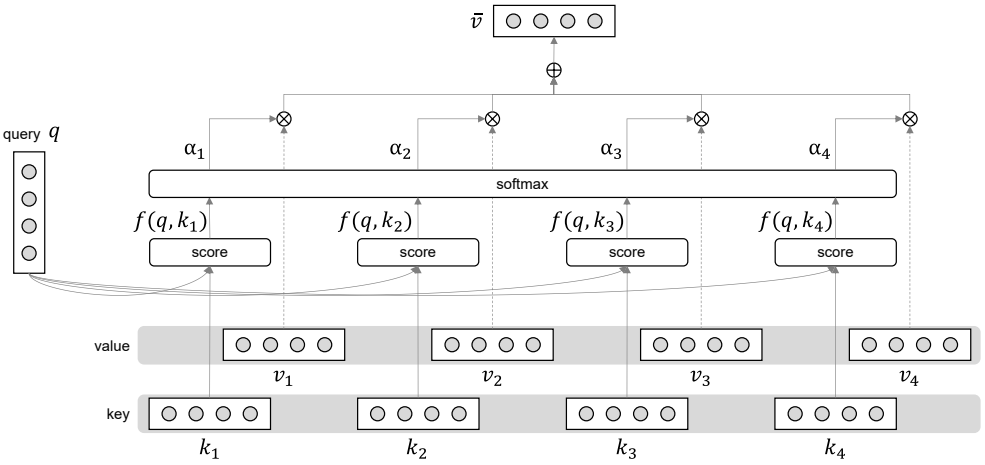

Figure 12. Attention mechanism

of the series. As shown in Figure 14, a score calculation is performed on all tokens, with each token as a query. Self-attention has the advantage over CNNs and RNNs in that it is easier to take into account the features of distant tokens. On the other hand, it has the disadvantage that the computational complexity of attention is proportional to the square of the series length.

The Transformer consists of the following four modules.

1. multi-head attention The Transformer architecture has three types of attention modules: (1) self-attention in the encoder, (2) self-attention in the decoder, and (3) attention between the encoder and decoder, called *cross-attention*. Each attention extracts multiple features by using multiple projections, called *multi heads*.

$$\begin{aligned} \mathrm{MultiHead}(Q,K,V) &= \mathrm{Concat}(\mathrm{head}_1,\cdots,\mathrm{head}_H)W^O, &(20)\\ \mathrm{head}_h &= \mathrm{Attention}(QW_h^Q, KW_h^K, VW_h^V), &(21) \end{aligned}$$

where $W_h^Q \in \mathbb{R}^{d_{\mathrm{model}}\times d_h}$, $W_h^K \in \mathbb{R}^{d_{\mathrm{model}}\times d_h}$, $W_h^V \in \mathbb{R}^{d_{\mathrm{model}}\times d_h}$, $W^O \in \mathbb{R}^{Hd_h\times d_{\mathrm{model}}}$, d_{model} is the number of dimension of the embedding vector, and H is the number of heads, respectively.

2. position-wise feed-forward networks The following equation explains the architecture of the position-wise feed-forward networks contained in

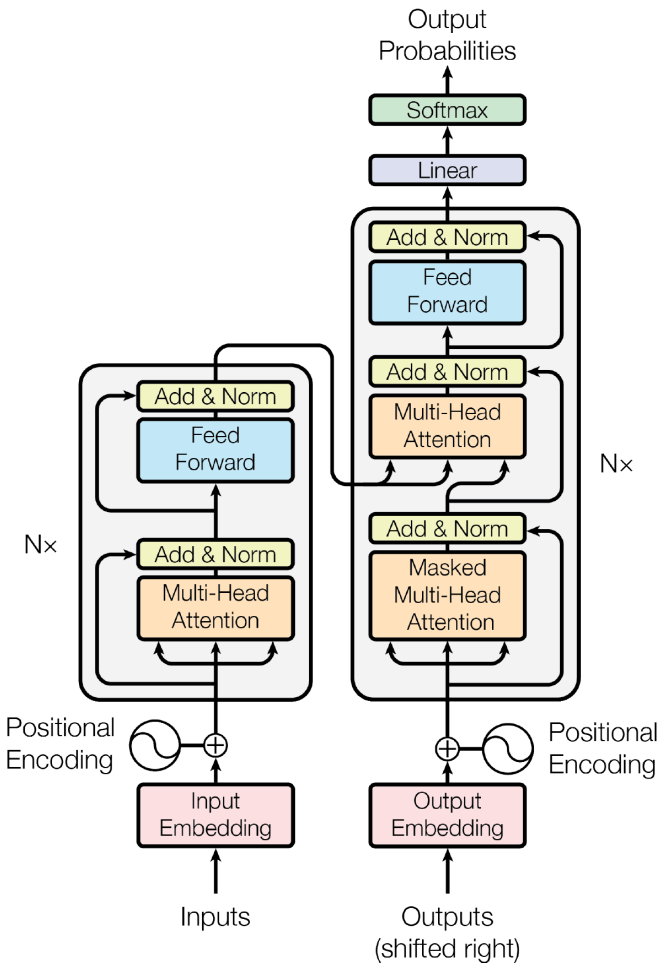

Figure 13. The Transformer - model architecture in [92]

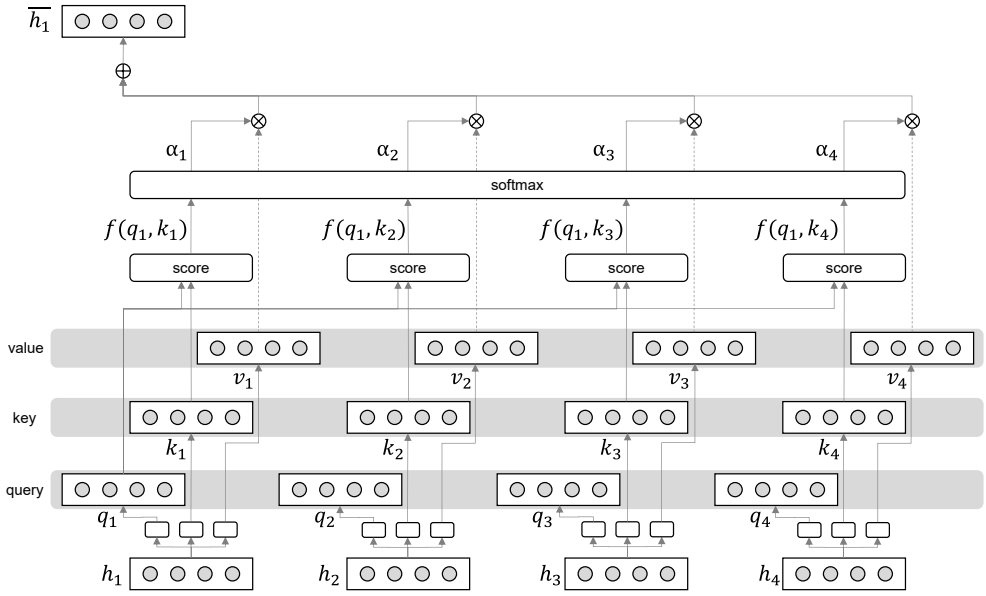

Figure 14. Self-attention mechanism

the encoder and decoder:

$$\mathrm{FFN}(x_t) = \mathrm{ReLU}(x_t W_1 + b_1)W_2 + b_2, \tag{22}$$

where $W_1 \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{ff}}}$, $b_1 \in \mathbb{R}^1 \times d_{\mathrm{ff}}$, $W_2 \in \mathbb{R}^{d_{\mathrm{ff}} \times d_{\mathrm{model}}}$, $W^O \in \mathbb{R}^{Hd_h \times d_{\mathrm{model}}}$, $b_2 \in \mathbb{R}^1 \times d_{\mathrm{model}}$, ReLU indicates the rectified linear unit [62], and d_{ff} indicates the number of intermediate layer dimensions in the feed-forward networks, respectively. It has been found that performance can be improved by increasing the dimensions in the intermediate layer d_{ff} above the original dimensions d_{model}.

3. residual connection and layer normalization Residual connection and layer normalization are applied to all attention and feed-forward networks, as shown in Figure 13.

$$x_t^{(l+1)} = \mathrm{LayerNorm}(x_t^{(l)} + f(x_t^{(l)})), \tag{23}$$

$$\mathrm{LayerNorm}(x_t) = \frac{g}{\sigma_t} \odot (x_t - \mu_t) + b, \tag{24}$$

$$\mu_t = \frac{1}{d_{\mathrm{model}}} \Sigma_{i=1}^{d_{\mathrm{model}}} x_{t,i}, \tag{25}$$

$$\sigma_t = \sqrt{\frac{1}{d_{\mathrm{model}}} \Sigma_{i=1}^{d_{\mathrm{model}}} (xt, i - \mu_t)^2}, \tag{26}$$

where l indicates the index of the layer. Layer normalization has the effect of preventing vector values from becoming extremely large.

4. positional encoding Since the attention itself does not consider ordering information, the same value may be obtained even if the order of the tokens is swapped. Therefore, position encoding, which expresses position information, is added to the embedded vector.

$$\mathrm{PE}_{(t,2i)} = \sin\left(t/10000^{2i/d_{\mathrm{model}}}\right), \tag{27}$$

$$\mathrm{PE}_{(t,2i+1)} = \cos\left(t/10000^{2i/d_{\mathrm{model}}}\right), \tag{28}$$

where $0 \leq i < d_{\mathrm{model}}/2$. An improved version of position encoding is to optimise using training data instead of fixed embedding by trigonometric functions [14]. Another approach is to add a bias to the self-attention score according to the relative distance between tokens rather than a vector corresponding to the absolute distance between tokens [78].

2.1.4 Delexicalization

Figure 15. Examples of delexicalization

In some cases, the values of specific data attributes constantly appear in the generated text as they are. In that case, the idea of template generation can be used. Instead of directly generating the data value, a symbol indicating that it is the attribute is generated and then replaced with the original value. This method is called *delexicalization* (Figure 15). This technique significantly reduces generation errors, as only symbols are to be estimated, regardless of the size of

the value variation. Furthermore, dealing with values not included in the training data (out of vocabulary (OOV)) is easy.

2.2 Controllable Data-to-Text

For multiple data inputs, data-to-text should take care of the input order. Usually, a data-to-text method consists of two processing modules: *content planning* [70, 43, 52, 16, 57, 82, 98, 28, 89, 36, 77, 30, 101, 50, 39, 55, 32, 31, 23, 59], which determines the order in which the data will be converted into sentences, and *surface realization*, which generates sentences from the results of content planning (Figure 16 (A)). As explained in the previous section, surface realization is mainly achieved using a language model, so that we will consider content planning here. Content planning is a module that reorders multiple data sets into the most appropriate order statistically determined from the training data. However, the order obtained by content planning is retained internally and passed to surface realization without being disclosed to the outside of the NLG model (Figure 16 (B)). For example, Kasner et al. [43] proposed a method that generates a text passage from resource description framework (RDF) triples in a zero-shot setting. RDF triples are converted using a template to a set of sentences, which are ordered to maximize their coherency by sentence ordering. However, this method automatically determines the sentence order without directly controlling the order using explicit constraints. Therefore, in order to have order controllability, the results of content planning must be accessible from the outside and must be able to be modified (Figure 16 (C) or (D)).

In recent years, several papers have been published addressing the data-to-text content order problem. Leng et al. [52] proposed a data-to-text method that controls sentence splits, entity order, and text length. They controlled the entity order by aligning the input data with the reference text. However, their method's performance in such standard automatic measures as BLEU [67], ROUGE_L [54], and METEOR [13] was lower than their baselines. Su et al. [82] proposed a data-to-text method that first predicts the suitable order of attributes from the data input and then generates output sentences from both the data input and the predicted order. Their method explicitly outputs ordering information as a content plan so that the order can be edited arbitrarily. Their human evaluation showed

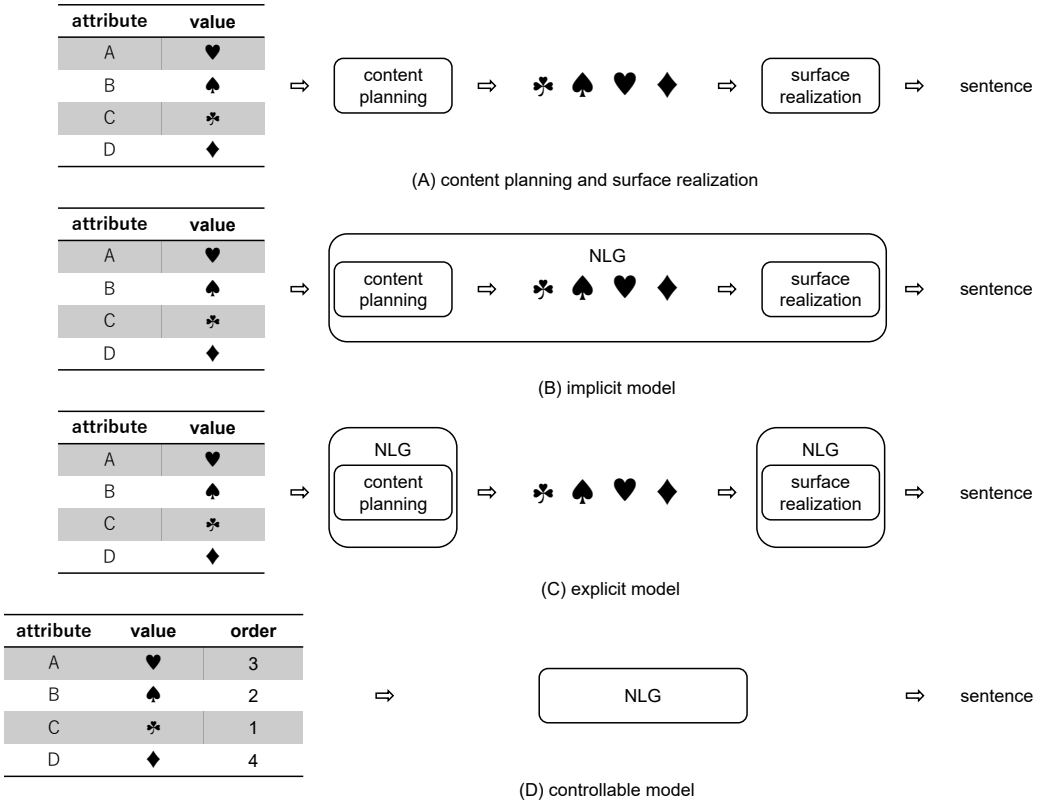

Figure 16. Content planning and surface realization

that the generated sentences accurately reflected the ordering information if the order was unedited. However, when the order was randomly shuffled, the accuracy was worse than without shuffling. Luo et al. [57] proposed a few-shot table-to-text generation method using a pre-trained language generation model. To generate sentences, the prompt "*summarize the following table:*" followed by the order of attributes that appear in the generated sentence and the input data are given to a pre-trained generation model. However, their human evaluation showed that the word order's accuracy was 84%, which is insufficient. Lin et al. [55] proposed a graph-based grouping planner. After the input data is encoded, it is picked up and grouped as data to be assigned to each sentence to be generated. The relationship among these groups is inferred to determine the order of the data in each sentence and the order of the groups. Although this method shows high performance for order estimation, it does not consider the possibility of freely controlling the order later since the order is learned so that the sentence is correctly reproduced in the training data. Xu et al. [98] proposed a method with explicit sentence planning, input ordering, and aggregation. In their method, a sentence planning module generates a graph of the input data. The module learns to align some input data with the target text using a hidden Markov model (HMM). For alignment, the target text is analyzed and transformed into sub-sentences containing subject, predicate, and object triples. As mentioned above, their method explicitly estimates the sentence plan, allowing the user to edit the order manually. However, their performance when the order changes still needs to be evaluated. Sasazawa et al. [76] proposed a method to control not only the content order but also the position of the content in the generated sentence. They use a relative position such as 0-10%, 10-20%, ..., 90-100%. In their methods, text length, keywords (data value), and keyword positions are encoded using an encoder. Then, the encoded vectors are decoded using a decoder. They reported that the accuracy of the position is 84% if the relative position is 0-10%. However, if the position is set to 30% or more, the accuracy is below 50%, which is far from practical.

Another controlled text generation method is *constrained decoding*, used in machine translation [35, 2, 69]. In this method, the words or phrases to be included in the translation are specified in advance, and only candidates con-

taining the specified words or phrases remain among the multiple candidate decoding results obtained by beam search. The application of this method to order-controllable data-to-text is as follows.

1. Determine the data values and order of appearance to be reflected in the generated text.
2. Set the output probability of tokens for values that should not yet be generated in the decoder to zero.
3. Output the token with the highest output probability.
4. Repeat steps 2 and 3 until all tokens corresponding to all values are output, and the special symbol token "$\langle eos \rangle$" is finally output.

However, the relationship between the data values and the generated text tokens must be predefined for this method to be applied to data-to-text. For instance, in the example shown in Table 1, the relationship between data values for the `name` and `near` attributes and text tokens corresponding to the values is unique because their values are directly represented in the text. If all the data consists of only these two attributes, the ordering control using this method is possible. On the other hand, for the value "*no*" for the `family friendly` attribute, it is challenging to uniquely determine in advance which part of the text corresponds to that value. For example, it could be three words: "*not kid friendly*" or one sentence: "*Unfortunately, it is not kid friendly.*" Many other possible texts correspond to the value, such as "*It is not family-friendly*" or "*is for adults only.*" In order to apply this method to order-controllable data-to-text, it is necessary to clarify all these relationships in advance. This is equivalent to creating a dictionary of all texts corresponding to the data input in advance, which is extremely difficult. Therefore, it is difficult to say that this method can be applied to any data.

2.3 Metrics for Automatic Evaluation

As metrics for automatic evaluation of the generated sentences, *word-overlap metrics* [20] such as BLEU [67], NIST [15], METEOR [13], ROUGE_L [54], and CIDEr [93] are commonly used. These metrics have strengths and weaknesses

and focus on different aspects of the generated text. However, no single metric can perfectly capture all aspects of text quality, such as relevance, fluency, and grammatical correctness. Thus, we usually use a combination of these metrics to evaluate the generated sentence.

We can also evaluate the generated sentences' fluency using pre-trained language models such as *perplexity* and *masked language model scoring*.

2.3.1 Word-Overlap Metrics

BLEU BLEU (Bilingual Evaluation Understudy) [67] is a metric obtained as the harmonic mean of the N-gram ($n \in 1, ..., 4$) precision of the system output concerning the human-produced reference sentences. If the length of the generated sentence is shorter than the reference sentence, it is lowered by a brevity penalty. The N-gram precision is the proportion of N-grams in the generated sentence that matches any reference sentence. Repeated N-gram matches are clipped to the maximum number of times the N-gram occurs in any single reference.

NIST NIST [15] is an extension of BLEU that also considers the informativeness of the N-grams, giving higher scores to less frequent N-grams (i.e., more informative) in the reference corpus.

METEOR METEOR (Metric for Evaluation of Translation with Explicit ORdering) [13] gauges the quality of machine translation by aligning the system output with individual human references. It measures both precision and recall of unigrams, and uses fuzzy matching based on stemming and WordNet synonyms, in addition to exact word matches. METEOR computes matches against multiple references separately and uses the best-matching one.

ROUGE_L ROUGE (Recall-Oriented Understudy for Gisting Evaluation)_L [54] is based on the longest common subsequences (LCS) between the system output and the human references, where a common subsequence requires the exact words in the same order but allows additional, non-covered words in the middle of either sequence. The final ROUGE_L score is an F-measure based on maximum precision and maximum recall achieved over any human reference, where precision

and recall are computed as the length of the LCS divided by the length of the system output and the reference, respectively.

CIDEr CIDEr (Consensus-based Image Description Evaluation) [93] was primarily designed for generated image captions but also applies to NLG. CIDEr is computed as the average cosine similarity between the system output and the reference sentences on the level of N-grams ($n \in 1, ..., 4$). The importance of the individual N-grams is given by the Term Frequency Inverse Document Frequency (TF-IDF) measure, which weighs an N-gram's frequency in a particular instance against its overall frequency in the whole dataset.

2.3.2 Pre-Trained Language Model-based Metrics

Perplexity with GPT *Perplexity* is a metric derived from information theory used to evaluate the performance of probabilistic or statistical models. In natural language processing, perplexity is a common evaluation metric for language models. Mathematically, the perplexity (PPL) of a discrete probability distribution p is defined as:

$$\mathrm{PPL}(p) = 2^{H(p)}, \tag{29}$$

where $H(p)$ represents the entropy (in bits) of the distribution:

$$H(p) = -\sum_{x} p(x) \log_2 p(x). \tag{30}$$

In the context of language models, perplexity is defined as:

$$\mathrm{PPL}(W) = \exp(-\frac{1}{|W|} \sum_{i}^{|W|} \log p(w_i)), \tag{31}$$

where W denotes the sequence of input text tokens and w_i is the i-th token of W. Here, $-\frac{1}{|W|} \sum_{i}^{|W|} \log p(w_i)$ is interpreted as cross-entropy, and perplexity can be calculated using a cross-entropy loss.

Perplexity can be calculated using a pre-trained language model such as OpenAI GPT-2 [71]. It can also be used to evaluate the quality of sentences given a pre-trained language model and a sequence of generated text [42]. The computed perplexity value could, to some extent, reflect the quality of the input sequence,

assuming that the pre-trained language model encapsulates language knowledge and behaves appropriately.

Masked Language Model Scoring Language models such as GPT-2 [71] are pre-trained to maximize the conditional probability of predicting the next word from the word output so far. However, this formulation does not suit masked language models such as BERT [14] and RoBERTa [56] because these models are pre-learning models by predicting words bi-directionally. Therefore, pseudo-log-likelihood(PLL) scores [75] have been proposed to calculate the naturalness of sentences in these masked language models. The scores are calculated as the sum of the log-likelihoods of the conditional probabilities when each word is masked and predicted, unlike the scores of regular language models, which consider probabilities in order from the front.

$$\log P_{\mathrm{MLM}}(W) = \sum_{t=1}^{|W|} \log P_{\mathrm{MLM}}(w_t | W_{\setminus t}), \tag{32}$$

where w_t is a token and

$$W_{\setminus t} := (w_1, \ldots, w_{t-1}, w_{t+1}, w_{|W|}). \tag{33}$$

2.4 Issues

As described in Section 2.2, many studies have addressed content ordering control, the target of this paper, in the context of "content planning." However, most studies estimate the ordering to be internal and not freely controllable by the user. Moreover, some methods allow explicit ordering control, but their performance needs to be improved.

3. E2E Refined Dataset

3.1 Introduction

Data-to-text is one of the main tasks of natural language generation (NLG) from a structured input such as tables, knowledge graphs, and resource description frameworks (RDFs). Meaning representation (MR)-to-text is one of the data-to-text tasks where MR consists of a set of pairs of a short text passage and a corresponding MR with some attribute-value pairs as shown in Table 1. There are several well-known corpora of MR-to-text, Weather (generating weather reports from meteorological data) [4], RotoWire (generating summaries of sports matches from game statistics) [97], WikiBio (generating biography from Wikipedia infobox) [50] and so on. The E2E dataset [63] in a restaurant recommendation domain used in the E2E NLG Challenge [20] is one of the most popular datasets for MR-to-text. However, this dataset was developed by crowdsourcing and suffers from errors in MR-text pairs that affect the performance of MR-to-text models. In this chapter, we aim to refine the E2E dataset by resolving errors and giving extra annotations. We fix errors in MR-text correspondences and remove inappropriate data samples from the dataset. We also provide additional annotations: *MR order*, *Number of sentences*, and *Sentence indexes* to control the generated text more precisely. In particular, MR order is the most important information for this paper's subject. We have developed this refined dataset to create an error-free MR-to-text dataset containing this information. We demonstrate that the refined dataset, called *E2E Refined Dataset* [88], improves MR-to-text performance.

3.2 E2E Dataset

The E2E dataset is made up of pairs consisting of a British English sentence and a corresponding MR with eight attributes: `name`, `eatType`, `food`, `priceRange`, `customer rating`, `area`, `familyFriendly`, and `near`, as shown in Table 1. However, some MR-text pairs contain deletion, insertion, and substitution errors. For example, in the text part of Table 2, the value "English" for the `food` attribute is missing, the value "city centre" for the `area` attribute is wrongly added, and the value "moderate" for the `priceRange` attribute is wrongly replaced with "cheap".

Table 2. Example of MR errors in the E2E dataset: **Bold** indicates deletion error, Underline indicates insertion error, and *Italic* indicates substitution error.

MR	`name`	The Punter
	`eatType`	coffee shop
	`food`	**English**
	`priceRange`	*moderate*
	`customer rating`	1 out of 5
	`area`	(empty)
	`familyFriendly`	yes
	`near`	Café Sicilia
Text	The Punter is a *cheap* family friendly coffee shop located in City Centre near Café Sicilia. 1 out of 5 customer rating.	

To properly control the content of a sentence in MR-to-text, it is necessary to exclude such incorrect data from the dataset. Although there have been some updates on the E2E dataset to resolve errors, including the cleaned [18] and enriched versions [24], these updated datasets still contain deletion, insertion, and substitution errors. In fact, we found a certain number of errors, which are shown in Table 3. To address this, we fixed errors in the MR-text correspondences and removed inappropriate data samples. We also refined the E2E dataset by manual annotations of the MR values to provide further constraints given by the text part.

The resulting E2E refined dataset consists of 40,560 examples for training, 4,489 for validation, and 4,555 for testing. Table 4 shows an example from it.

3.3 Text Refinement

The E2E dataset contains errors and discrepancies in the language used. We improved the quality of the text parts of the dataset by rectifying errors and standardizing expressions as follows.

Table 3. Number of MR labelling errors in each dataset (Original: E2E dataset[63], Cleaned: Cleaned dataset [18], Enriched: Enriched dataset [24]).

Error type	Dataset	Training	Validation	Test
Deletion	Original	10,931	1,096	1,315
	Cleaned	23	1	1
	Enriched	1,262	145	89
Insertion	Original	10,028	263	16
	Cleaned	4,475	471	745
	Enriched	25,570	2,724	3,082
Substitution	Original	9,290	794	945
	Cleaned	5,795	616	666
	Enriched	4,172	413	395

3.3.1 Error Correction

We refined various types of errors in the original E2E dataset. We focused on the following four error categories.

Irregular MR Values An MR should only contain one value or be left empty for each attribute. However, some MR data contains two values for the `name` and `near` attributes. To ensure accuracy, we removed any irrelevant data from the dataset. (e.g. *The **Cotto** ranges between twenty and twenty-five pounds with a high customer rating. **The Portland Arms** serves Italian food near the riverside.* → (removed))

Overlaps We removed duplicated phrases within a sentence. (e.g. *The Golden Curry served English food, **is adult only**, is in the city centre, **is adult only**, has a customer rating of 5 out of 5 and is near the Café Rouge.* → *THE GOLDEN CURRY served English food, **is adult only**, is in the city centre, has a customer rating of 5 out of 5 and is near the CAFÉ ROUGE.*)

Indefinite Articles We fixed any errors in the usage of the indefinite articles "a" and "an." (e.g. *For **an** child friendly, average coffee shop serving fast food*

Table 4. Example of the E2E refined dataset: Original sample of the E2E dataset is shown in Table 1.

	Attribute	Value	Order	Sentence index
MR	`name`	NAME (THE WRESTLERS)	1	1
	`eatType`	restaurant	3	1
	`food`	Italian	4	1
	`priceRange`	moderate	2	1
	`customer rating`	(empty)	0	0
	`area`	city centre	5	2
	`familyFriendly`	yes	7	3
	`near`	NEAR (RAJA INDIAN CUISINE)	6	2
# sentences	3			
Text	THE WRESTLERS is a moderately priced restaurant that serves Italian food. It is located in the city centre near RAJA INDIAN CUISINE. Great place to bring your family.			
Text (delexicalized)	NAME is a moderately priced restaurant that serves Italian food. It is located in the city centre near NEAR. Great place to bring your family.			

*try The Eagle, riverside near Burger King. → For **a** child friendly, average coffee shop serving fast food try THE EAGLE, riverside near Burger King.*)

Typos We identified and fixed more than 3,700 typographical errors in the MR values and sentences. (e.g. *A kid friendly Strada has moderate price range and customer rating of 1 **of of** 5. → A kid friendly STRADA has moderate price range and customer rating of 1 **of** 5.*)

3.3.2 Normalization

We normalized the text in the following six types of aspects.

British English Since the E2E dataset is based on British English, we substituted such words with American spellings as "center", "flavor", "organize", and "traveling" with "centre", "flavour", "organise", and "travelling." (e.g. *Alimentum providing fast food less than £20 price range. It is located in city **center**. → ALIMENTUM providing fast food less than £20 price range. It is located in city **centre**.*)

Prices `priceRange` is categorized into "cheap", "moderate", "expensive", "lower than £20", "£20-25", and "more than £30" as shown in Table 5. In that case, "£23" should be labelled as "£20-25". However, to eliminate confusion, we used the label "£20-25" for all prices falling within that range, including values like "£23", "£22", "£24", and "from £20 to £25". This approach was also taken for the labels "lower than £20" and "more than £30". (e.g. *If you're looking for pub grub or Indian food, you could try The Plough. No you can't take your kids there but the prices are reasonable about **£24** for a meal. You'll find it near to Café Rouge. → If you're looking for pub grub or Indian food, you could try THE PLOUGH. No you can't take your kids there but the prices are reasonable about **£20-25** for a meal. You'll find it near to CAFÉ ROUGE.*)

Currency Expressions For ease of use, we standardized the currency unit as "£20" instead of using variations such as "20 pounds", "20gbp", "20lb", "20 quid" and so on. (e.g. *Close to Yippee Noodle Bar in the city centre a French*

Table 5. All variations of MR values in the E2E refined dataset.

Attribute	# variations	MR values (delexicalized)
`Name`	1	NAME
`eatType`	4	(empty), coffee shop, pub, restaurant
`food`	11	(empty), American, Canadian, Chinese, English, fast food, French, Indian, Italian, Japanese, Thai
`priceRange`	7	(empty), £20-25, cheap, expensive, less than £20, moderate, more than £30
`customer rating`	7	(empty), 1 out of 5, 3 out of 5, 5 out of 5, average, high, low
`area`	3	(empty), city centre, riverside
`familyFriendly`	3	(empty), no, yes
`near`	2	(empty), NEAR

*restaurant, Alimentum, has low ratings but the price is less than **20lb**. → Close to YIPPEE NOODLE BAR in the city centre a French restaurant, ALIMENTUM, has low ratings but the price is less than **£20**.*)

Symbols We normalized such symbols as periods, commas, white spaces, etc. (e.g. *Browns Cambridge sells Indian food, and is kids friendly**,,** it is in riverside area near The Sorrento → BROWNS CAMBRIDGE sells Indian food, and is kids friendly**,** it is in riverside area near THE SORRENTO.*)

Quotation Marks We used single quotation marks instead of double quotations. (e.g. *A highly rated coffee shop **"**The Punter**"** serving English food priced between £20 - £25 and is child friendly. → A highly rated coffee shop **'**THE PUNTER**'** serving English food priced between £20-25 and is child friendly.*)

Capital Letters We fixed the capitalization errors in proper nouns and words that begin sentences. (e.g. ***a** coffee shop Zizzi located by the riverside has a high price range with an average customer rating. **t**hey are children friendly → **A** coffee shop ZIZZI located by the riverside has a high price range with an average*

*customer rating. **T**hey are children friendly.*)

3.4 MR Refinement

The MRs in the original E2E dataset include labelling errors. We refined the MR labels as described in Section 3.4.1. We provided additional annotations for further controllable generation study regarding flexible content planning, as described in Section 3.4.2.

3.4.1 Labelling

Throughout the E2E dataset, we corrected MR labelling errors manually. Additionally, we replaced the value "high" with "expensive" for the `priceRange` attribute. Moreover, we added new labels for the `food` attribute, including "American", "Canadian", "Indian", and "Thai". Table 5 lists all the refined labels.

3.4.2 Additional Annotations

MR Order We marked the order of the MR values mentioned in the corresponding sentences, as shown in Table 4. In case of an empty MR value, we represented the order with a "0".

Number of Sentences In Table 4, we indicated the number of sentences present in the text. We identified the number of sentences by looking for periods (".") and question marks ("?"). As per the example given in the table, the text portion contains three periods, so we assigned the number of sentences as "3".

Sentence Indexes We also provided annotations for the appearance of each MR value in the corresponding sentences as shown in Table 4. For instance, the values of `eatType`, `area`, and `familyFriendly` are found in the first, second, and third sentences, respectively. In cases where an MR value is empty, we set the index to "0".

3.5 MR-Text Pair Refinement

To further enhance the dataset, we refined the MR-text pairs by eliminating repetitions and utilizing a strategy to convey certain values effectively in both the MR and text.

3.5.1 Deduplication

We excluded approximately 1,500 MR-text pairs from the dataset as a result of the deduplication process.

3.5.2 Delexicalization

Since all of the `name` and `near` values appear as-is in the sentences, we replaced such values in the text and MR values with "NAME" and "NEAR" to standardize the data. We stored the values in uppercase to preserve the original information, although the standardized forms are still useful for training MR-to-text models.

3.6 Experiments

We investigated the effect of the refinement by the following experiments.

3.6.1 Dataset

We used the original E2E dataset and the E2E refined dataset. Here, we capitalized only the first letter of each word of `name` and `near` values and did not replace the value "high" with "expensive" for the `priceRange` attribute for the E2E refined dataset, for a fair comparison. We did not use additional annotations (described in Section 3.4.2) either.

3.6.2 Method

We used TGEN[1] [19], based on an LSTM-based sequence-to-sequence model with an attention mechanism. TGEN was used as the baseline for the E2E NLG Challenge [20]. The detail of this method is described in Section 4.4.1. We

[1] `https://github.com/UFAL-DSG/tgen`

trained two models using the training sets of both the original E2E dataset and the E2E refined dataset, respectively.

3.6.3 Metrics

To assess the performance of both models, we utilized several evaluation metrics; BLEU [67], NIST [15], METEOR [13], ROUGE_L [54], and CIDEr [93] (described in Section 2.3.1). These metrics can be obtained through the E2E challenge metrics script[2]. The evaluation was conducted on the test set of the E2E refined dataset.

3.6.4 Results

Table 6 shows the scores. The model's performance trained using the E2E refined dataset outperformed that of the model using the original dataset, except for ROUGE_L. These results suggest that the refined dataset contains more accurate label information, which ultimately led to improved performance.

Table 6. Results of automatic evaluations

Dataset	BLEU(↑)	NIST(↑)	METEOR(↑)	ROUGE_L(↑)	CIDEr(↑)
E2E original	0.5462	7.6209	0.4103	0.6561	2.2448
E2E refined	0.5581	7.8378	0.4252	0.6488	2.3865

3.7 Limitations

Although we modified the E2E dataset for the development of MR-to-text models, the following limitations remain:

- As mentioned in Section 3.3.1, we removed the data that had irregular MR values. However, multiple values may be allowed under different formulations of MR-to-text problems for more complex situations.
- We currently ignore referring expressions, although they should be allowed generally.

[2] `https://github.com/tuetschek/e2e-metrics`

- We regard attributes other than `name` as modifiers of a `name`. However, an attribute sometimes modifies `near`. Our current formulation ignores such relationships.

3.8 Conclusion

In this chapter, we described our E2E refined dataset. We reduced errors in the dataset by correcting mistakes and normalizing some expressions to make the sentences simpler. Additionally, we refined the annotation of the MR values by annotating the MR order, the number of sentences, and the sentence indexes as additional information. Our experimental results showed that this refined dataset led to improved performance of NLG. The dataset and data conversion programs in Python are available here[3]. We hope that this dataset will be useful for future research in related fields.

[3] `https://github.com/KSKTYM/E2E-refined-dataset`

4. Content Order-Controllable MR-to-text

4.1 Introduction

In this chapter, we propose an MR-to-text (MR2T) method that controls the order of the MR values in generated text passages with additional value order annotations using the E2E refined dataset described in Chapter 3. First, we train an order-constrained MR2T model with a text-to-MR (T2MR) model. Then, we augment the MR-text dataset to obtain a better-balanced distribution in terms of the number of non-empty attributes by synthesizing the attribute-value pairs that do not appear in the dataset. We investigate the performance of an MR2T model trained using the augmented dataset by automatic and subjective evaluations and show that it can control the order of the MR values with high accuracy [87]. Our code is available at here[4].

4.2 Dataset

The E2E dataset [63] used in the E2E NLG Challenge [21] is widely utilized in MR2T studies. It consists of a set of pairs of a British English text passage and a corresponding MR with the following eight attributes in a restaurant recommendation domain: `name`, `eatType`, `food`, `priceRange`, `customer rating`, `area`, `familyFriendly`, and `near`. However, some of its MR-text pairs suffer from the following errors:

- *deletion*: some attribute-value pairs are not mentioned in the text;
- *insertion*: some empty attributes are mentioned incorrectly with unintended values;
- *substitution*: some MR values are replaced by wrong ones.

Such errors must be fixed to properly control a text passage's content by MR2T. Although updates have rectified some of these errors [18, 24], the updated datasets still include such errors. We refined the E2E dataset with additional error fixes by manual annotations of the MR values and made it public as the

[4] `https://github.com/KSKTYM/content_order-controllable_mr-to-text`

E2E refined dataset [88]. It also includes the order of the MR values in the corresponding text passage (Table 7). We used the E2E refined dataset in this work to train the MR2T models.

Table 7. Example of E2E refined dataset

	Attribute	Value	Order
MR	`name`	NAME (WILDWOOD)	1
	`eatType`	restaurant	2
	`food`	Italian	3
	`priceRange`	(empty)	0
	`customer rating`	(empty)	0
	`area`	riverside	5
	`familyFriendly`	no	6
	`near`	NEAR (RANA INDIAN CUISINE)	4
Text	NAME(WILDWOOD) is a restaurant that serves Italian food located near NEAR(RAJA INDIAN CUISINE) in the area of riverside. Unfortunately, it is not kid friendly.		

4.3 Data Augmentation

The number of training instances with non-empty attributes is shown in the second column of Table 8. For example, the number of training instances with eight non-empty attributes is only 1,190; that with five non-empty attributes is 12,442. Such an imbalanced training data distribution causes poor performance for instances with many non-empty attributes, as shown later in the experimental results.

Data augmentation is a promising approach for mitigating such data imbalance. Existing studies on data augmentation for NLG use text generation and text analysis models. Kedzie et al. [45] used noise injection sampling. First, they synthesized the under-represented MR in the training data. Second, they converted the MR to a text passage using their MR2T model by injecting Gaussian noise into the decoder hidden states. Then they obtained an MR using their

Table 8. Number of training data in terms of non-empty attributes: NEA denotes non-empty attributes.

# NEA	Original data	Augmented data	Merged data
1	55	0	55
2	393	7	400
3	2,910	926	3,836
4	8,119	4,323	12,442
5	12,442	0	12,442
6	10,058	2,384	12,442
7	5,393	7,049	12,442
8	1,190	11,252	12,442
Total	40,560	25,941	66,501

Table 9. Number of every possible combination of MR values and MR orders and obtained samples: NEA denotes non-empty attributes.

# NEA	MR value	MR order	MR value&order	Obtained samples
1	1	1	1	0
2	30	2	60	11
3	355	6	2,130	1,167
4	2,142	24	51,408	30,949
5	7,096	120	851,520	112,139
6	12,912	720	9,296,640	206,353
7	11,952	5,040	60,238,080	191,212
8	4,320	40,320	174,182,400	69,120
Total	38,808	-	244,622,239	610,955

MR parser. Finally, a pair of the obtained MR and the generated text passage was accepted as augmented data. Unfortunately, noise injection sampling made insertion and deletion errors in the generated text passage.

Chai et al. [7] proposed a feedback-aware self-training method for their conditional text generation. First, they generated a text passage whose condition is different from the original. Then a classifier predicted the condition from the generated text passage. A condition-passage pair was used as augmented data if the input and predicted conditions matched.

We applied an idea that resembles Chai's approach to augment the training data with synthetic examples generated in the following steps.

Step 1 Generate every possible combination of MR values and orders. The number of MR value patterns is calculated from the variation of the MR values (Table 5). The number of MR order patterns is a factorial of their non-empty attributes. Since the number of generated combinations is enormous (244.6 million), we randomly sampled them to 16 patterns for an MR order, removed the MRs included in the original dataset to avoid data leakage, and obtained 610,955 MR combinations (Table 9).

Step 2 Convert the MRs obtained in Step 1 to text passages by the MR2T model trained using the original training data. The model's details are explained in Section 4.4.4.

Step 3 Convert the text passages obtained in Step 2 to MRs using the T2MR model trained using the original training data. The T2MR model is also explained in Section 4.4.4.

Step 4 Augment the training data with the pairs of an MR and a text passage as *synthetic MR-text pairs* when the result of Step 3 matches the MR generated in Step 1. Here our motivation is to balance the data distribution. We sampled the pairs for a total of 12,442 (the maximum number in the original training data with five non-empty attributes) for each non-empty attribute (Table 8). Finally, we obtained 66,501 MR-text pairs as the augmented training data.

4.4 Method

Next we explain the MR2T methods used in this work: TGEN [19], JUG [90], Transformer-based MR2T, and the T2MR methods based on Transformer.

4.4.1 TGEN

As our baseline, we use TGEN [19], based on an LSTM-based sequence-to-sequence model with an attention mechanism. The system takes in dialogue acts (DA) as input, representing a trio of "DA type, slot, value" for each attribute. The "DA type" is always set to "inform" because the E2E dataset only contains one action. The MR2T generator, equipped with an attention mechanism, uses an LSTM (Long Short-Term Memory) network in its encoder to transform an input sequence $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, where each x_i represents an input token, into a series of encoder outputs and hidden states $\boldsymbol{h} = \{h_1, \ldots, h_n\}$, where $h_t = \text{LSTM}(x_t, h_{t-1})$ is a non-linear function represented by the LSTM cell. The decoder then uses these hidden states to generate a sequence $\boldsymbol{y} = \{y_1, \ldots, y_m\}$ with another LSTM-based RNN.

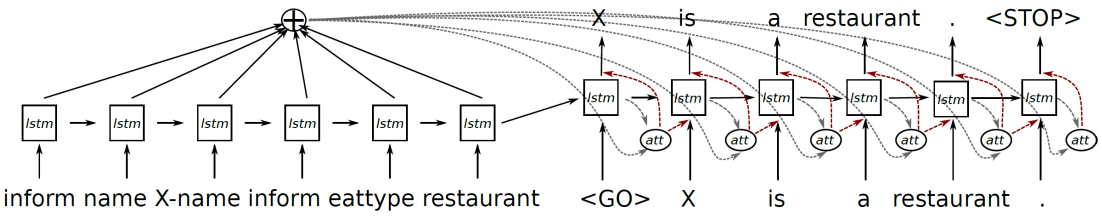

Figure 17. Sequence-to-sequence generator of TGEN [19]

When TGEN generates output sentences using beam search, it employs a classifier to rerank the top n outputs from the beam search. It penalizes those that either lack necessary information or include irrelevant ones. This classifier encodes the generated sentence, and the final hidden state of the encoder is used to compute the output 1-hot vector. The input DA is also transformed into a 1-hot vector. The reranker then computes the Hamming distance between these two vectors, which measures the number of positions at which the corresponding values are different.

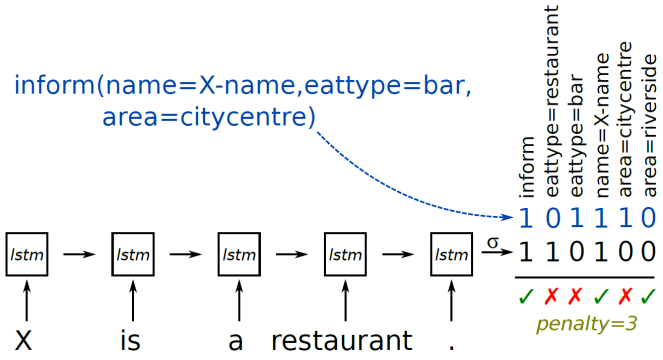

Figure 18. The reranker used in TGEN [19]

We used the distributed programs[5] "as-is". TGEN is also used as the baseline for the E2E NLG Challenge [21]. Note that TGEN does not constrain the order of the MR values in its text generation.

4.4.2 JUG

As another baseline, we use JUG [90], based on an LSTM-based generative model for joint natural language understanding (NLU) and generation, which couples NLU and NLG through a shared latent variable. In simpler terms, the input MR data x and the sentences generated y are associated with the same abstract latent vector z, representing the shared information between the input and output. Here, NLU is equivalent to T2MR, and NLG corresponds to MR2T.

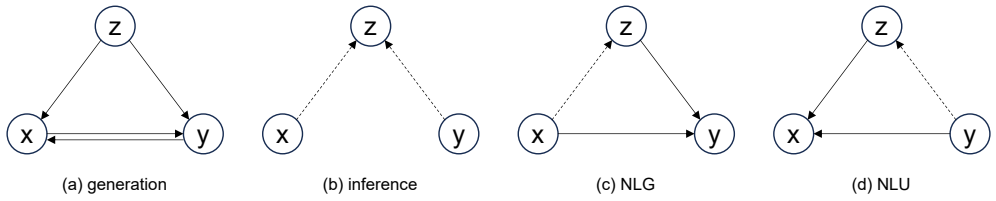

Figure 19. Generation and inference process in JUG, and how NLU and NLG are achieved [90]. x and y denotes MR and sentences respectively; z represents the shared latent vector for x and y.

During generation, this abstract latent vector guides the standard conditional

[5] `https://github.com/UFAL-DSG/tgen` (distributed under the Apache License 2.0)

generation of either NLG or NLU (shown in Figure 19 (a)). The abstract vector can be inferred from either the MR data or the sentences (shown in Figure 19 (b)). Therefore, for NLG, the abstract latent vector should be inferred from the MR data. Then, the generated sentence is derived from both the MR data and the abstract latent vector (shown in Figure 19 (c)). The chosen posterior distribution $q(z|x)$ is Gaussian. The task of inferring z is then reframed as computing the mean μ and standard deviation σ of the Gaussian distribution, which represents the central tendency and dispersion of the distribution, respectively, using an NLG encoder. As the encoder, a bi-directional LSTM is used, which computes a list of hidden vectors H, each representing the concatenation of forward and backward LSTM states. These hidden vectors are average-pooled and passed through two feed-forward neural networks to compute the $\boldsymbol{\mu}_{x,z}$ and $\boldsymbol{\sigma}_{x,z}$ vectors of the posterior $q(z|x)$.

$$\boldsymbol{H} = \text{Bi-LSTM}(x) \tag{34}$$

$$\bar{\boldsymbol{h}} = \text{Pooling}(\boldsymbol{H}) \tag{35}$$

$$\boldsymbol{\mu}_{x,z} = \boldsymbol{W}_{\mu}\bar{\boldsymbol{h}} + \boldsymbol{b}_{\mu} \tag{36}$$

$$\boldsymbol{\sigma}_{x,z} = \boldsymbol{W}_{\sigma}\bar{\boldsymbol{h}} + \boldsymbol{b}_{\sigma}, \tag{37}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ denote neural network weights and bias. The latent vector z can then be sampled from the approximated posterior using the re-parameterization trick:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}) \tag{38}$$

$$\boldsymbol{z} = \boldsymbol{\mu}_{x,z} + \boldsymbol{\sigma}_{x,z}\boldsymbol{\epsilon} \tag{39}$$

To generate sentence y based on latent variable z and MR data x, an LSTM decoder that relies on both z and x via an attention mechanism is used. At each time step, the decoder calculates:

$$\boldsymbol{g}_i^y = \text{LSTM}(\boldsymbol{g}_{i-1}^y, \boldsymbol{y}_{i-1}) \tag{40}$$

$$\boldsymbol{c}_i = \text{attention}(\boldsymbol{g}_i^y, \boldsymbol{H}) \tag{41}$$

$$p(y_i) = \text{softmax}(\boldsymbol{W}_v[\boldsymbol{c}_i \oplus \boldsymbol{g}_i^y \oplus \boldsymbol{z}] + \boldsymbol{b}_v \tag{42}$$

where $\oplus$ signifies concatenation, $\boldsymbol{y}_{i-1}$ is the word vector of input token; $\boldsymbol{g}_i^y$ is the corresponding decoder hidden state and $p(y_i)$ is the output token distribution at time step i.

According to [90], the BLEU score for the original E2E dataset was better than that of TGEN. We use the distributed programs[6] "as-is". This method does not constrain the order of the MR values in its text generation either.

4.4.3 Transformer without Order Constraints

We use Transformer [92] for MR2T, configured as shown in Figure 20. The Transformer-based MR2T model takes a sequence of MR values as its input tokens and generates output tokens one by one. For example, it takes ["⟨*sos*⟩", "*NAME*", "*restaurant*", "*Italian*", "*riverside*", "*no*", "*NEAR*", and "⟨*eos*⟩"] as the input tokens when the MR shown in Table 7 is used. Here each MR value is treated as one token without further tokenization into subwords. "⟨*sos*⟩" and "⟨*eos*⟩" are special symbol tokens that express a sequence's start and its end. "⟨*unk*⟩" is another special symbol token that express unknown tokens in the dataset. The MR-value tokens are given in a fixed order from `name` to `near`, and empty values are excluded from the input. Note that since all the MR values are unique (Table 5), the input tokens do not need to include MR attributes. For the positional values, we use [0, 1, ..., $n + 1$] (where n equals the number of non-empty attributes) as the input vector. These values are embedded with a trainable embedding.

For T2MR, we use Transformer with the same structure as that for MR2T. It takes a sequence of text tokens as input and predicts the MR values one by one in the fixed order of the attributes. We use the `word_tokenizer` module in the Python NLTK library for the text tokenization. For example, it takes ["⟨*sos*⟩", "*NAME*", "*is*", "*a*", "*restaurant*", ..., "*kid*", "*friendly*", "*.*", and "⟨*eos*⟩"] as input to induce the corresponding MR value sequence as output: ["⟨*sos*⟩", "*NAME*", "*restaurant*", "*Italian*", "*riverside*", "*no*", "*NEAR*", and "⟨*eos*⟩"].

For the inferences, MR2T runs greedily to generate a text passage, and T2MR predicts its MR values. If the T2MR result matches the original MR, the generated text passage is deemed reliable. If the MRs are not identical, MR2T generates text passages using beam search (width = 5) and applies T2MR to check whether the result is reliable and chooses the reliable one with the best

[6] `https://github.com/andy194673/Joint-NLU-NLG`

score.

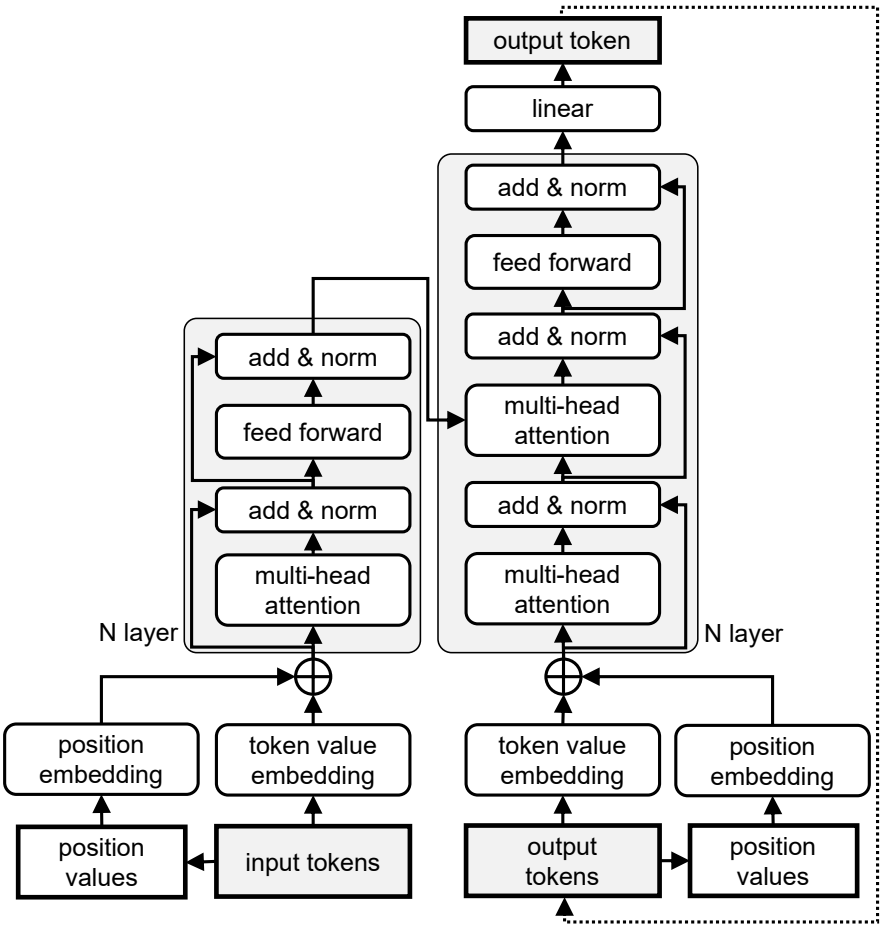

Figure 20. Transformer model

4.4.4 Transformer with Order Constraints

We propose another Transformer-based MR2T model that takes the content order constraints. For example, it takes [“⟨*sos*⟩”, “*NAME*”, “*restaurant*”, “*Italian*”, “*NEAR*”, “*riverside*”, “*no*”, and “⟨*eos*⟩”] as input, where MR-value tokens appear in the corresponding order with their mentions in the text passage. The order-constrained T2MR model also has identical Transformer architecture, although it is trained to predict the MR values in the order of their mentions in the input text. We train these models with the original and augmented training

datasets.

4.4.5 Transformer with Order Constraints in Decoder

We propose another Transformer-based MR2T model for order control. This method inputs the order control condition to the decoder, whereas the method in Section 4.4.4 inputs it to the encoder. In other words, the input tokens to the encoder are the same as the method described in Section 4.4.3, which does not input the order condition. MR order tokens are given before the special symbol token "$\langle sos \rangle$" for the input tokens to the decoder. For example, it takes ["*1*", "*2*", "*3*", "*5*", "*6*", "*4*", and "$\langle sos \rangle$"] as the input tokens when the MR shown in Table 7 is used. The MR-order tokens are given in a fixed order from `name` to `near`, and "0" is excluded from the input. This allows order control to be performed by the decoder. We train this model with the original training dataset.

4.5 Experiments

We investigated the effect of the order constraints in MR2T by the following experiments.

4.5.1 Data

We used the E2E refined dataset with 40,560 MR-text pairs for training and 4,555 pairs for evaluation. We also used the augmented training data (66,501 MR-text pairs) to train the Transformer with an order constraints model. To investigate the MR2T performance with different MR orders, we augmented the test data by reordering the MR values and used them as inputs. We sampled four different orders for the test data with three or more non-empty attributes and the alternative MR value order for those with two non-empty attributes and obtained 21,140 additional test instances. We then used four random seeds for the test data augmentation and obtained four sets of augmented test data (21,140×4 sets). Hereafter, we call them the *reordered test data*. Here all of the `name` and `near` values are delexicalized for all models described in Section 4.4, because they appear as-is in the text passages. The values are restored after generating sentences by MR2T models.

Note that all MR values in the E2E refined dataset used in this experiment (see Table 5) are included in the training data. Hence, no unknown MR values are input during the validation and evaluation of the models in this experiment. In addition, the models are designed for only the eight types of attributes described in Section 3.2. Therefore, if attributes or MR values not included in the training data are to be input to the models, another training data containing them must be prepared, and the models must be retrained.

4.5.2 Model Configuration

We set the embedding vector size to 256, the feed-forward network vector size to 512, the head number to 8, and the layer number to 3. For training, we used the following settings: a batch size of 128, a learning rate of 0.0005, 100 epochs, a dropout rate of 0.1, and a clip norm of 1.0. The model was optimized using Adam [47]. We used one NVIDIA GeForce RTX 3090 GPU. It took about 33 minutes to train the MR2T model with the original training data and 53 minutes to train it with the augmented training data. We chose the best model that resulted in minimum loss on the validation data in the E2E refined dataset among those 100 trained models at the end of the training epochs. The loss was saturated in ten epochs by the MR2T model without order constraints, nine epochs by that with the order constraints using the original training data, and ten epochs by that using the augmented training data. For TGEN and JUG, we used the distributed programs "as-is".

4.5.3 Metrics

We evaluated the MR2T performance by BLEU [67], NIST [15], METEOR [13], ROUGE_L [54], and CIDEr [93], all of which can be obtained using the E2E challenge metrics script[7] (described in Section 2.3.1). To calculate these metrics, we prepared the following two types of references for the test instances:

- *Order-independent references*: those corresponding to the given MR without order constraints (the average number of reference text passages: 6.67);

[7] `https://github.com/tuetschek/e2e-metrics`

- *Order-dependent references*: those corresponding to the given MR with order constraints (the average number of reference text passages: 1.21).

We also evaluated the performance by checking whether the original MR to MR2T and the predicted MR from T2MR are identical. We named this method *MRcheck*. Since the T2MR performance is highly accurate, as shown in Table 10, we believe it can be used to evaluate MR2T. We used the order-constrained T2MR model trained with the augmented training data for MRcheck.

Furthermore, we evaluated the fluency of the generated sentences by perplexity with GPT-2[8] and pseudo-log-likelihood (PLL) scores with BERT[9] described in Section 2.3.2.

Table 10. Accuracy of T2MR models. **Bold** indicates best result. (*: trained using augmented training data)

Method	MR value&order	MR value	MR order
Transformer w/o order	n/a	98.18	n/a
Transformer w/ order	98.62	98.95	98.84
Transformer w/ order(*)	**98.79**	**99.25**	**99.06**

4.5.4 Results

Table 11 shows the scores. The Transformer MR2T model without order constraints outperformed TGEN and JUG, indicating Transformer's effectiveness in this task. Our proposed order-constrained MR2T model clearly outperformed the model without order constraints. Comparing the order-constrained MR2T model where the order constraints are given to the encoder with the model where they are given to the decoder, the former shows higher performance. The proposed model trained using the augmented data also outperformed the model trained with the original data. Although an advantage was observed even with the order-independent references, it was much larger with the order-dependent references.

Table 12 shows the MRcheck results. The order-constrained Transformer preserved the MR order very accurately (99.58%) for the original test data, although

[8] `https://huggingface.co/docs/transformers/model_doc/gpt2`

[9] `https://github.com/awslabs/mlm-scoring`

Table 11. Results of automatic evaluation: **Bold** indicates best result for order-independent reference, and *Italic* indicates best result for order-dependent reference (I: order-independent reference, D: order-dependent reference, †: order constraints given in decoder, *: trained using augmented training data)

Method	Ref.	BLEU(↑)	NIST(↑)	METEOR(↑)	ROUGE_L(↑)	CIDEr(↑)
TGEN	I	0.5626	7.8907	0.4278	0.6614	2.4066
	D	0.3339	5.6815	0.3762	0.5262	2.0885
JUG	I	0.5733	7.6896	0.4337	0.6488	2.3972
	D	0.3505	5.6242	0.3871	0.5201	2.0580
Transformer	I	0.5840	7.8227	0.4384	0.6659	2.5141
w/o order	D	0.3600	5.7151	0.3910	0.5344	2.1753
Transformer	I	0.6280	8.6083	0.4595	0.7551	2.7783
w/ order	D	0.4836	7.0947	0.4356	0.7422	3.3062
Transformer	I	0.6232	8.6180	0.4562	0.7489	2.7571
w/ order(†)	D	0.4771	*7.1033*	0.4318	0.7352	3.2709
Transformer	I	**0.6335**	**8.6198**	**0.4624**	**0.7575**	**2.7855**
w/ order(*)	D	*0.4914*	7.0941	*0.4393*	*0.7453*	*3.3383*

the baselines failed to do so (6.74%, 5.93%, and 7.42%). Even though the accuracy of the reordered test data (88.07%) was worse than that for the original test data (99.58%), the performance of the proposed model trained using the augmented training data was almost perfect (99.95%).

Comparing the order-constrained MR2T model where the order constraints are given to the encoder with the model where they are given to the decoder, the former shows higher performance as listed in Table 11 and Table 12. In particular, the performance of MRcheck accuracy for the MR order of the latter model is substantially degraded. These results suggest that when order constraints are input to the decoder, the vectors generated by the encoder need to correspond to all order constraint variations, which increases the information to be embedded and results in lower performance.

Furthermore, we found more MRcheck errors in the test instances with a larger number of non-empty attributes (Table 13): for example, 12 and 1,055.8 errors for the instances with eight non-empty attributes for the original and reordered test data. However, the amount of proposed model training using the

Table 12. MRcheck results of accuracy: **Bold** indicates best result for original test data, and *Italic* indicates best result for reordered test data (†: order constraints given in decoder, *: trained using augmented training data)

Method	Test data	MR value & order	MR value	MR order
TGEN	original	6.74	99.41	6.74
JUG	original	5.93	98.24	5.93
Transformer w/o order	original	7.42	100.0	7.42
Transformer w/ order	original	99.58	99.96	99.58
	reordered	88.03	96.74	88.07
Transformer w/ order(†)	original	94.64	99.58	94.67
	reordered	56.10	94.87	56.14
Transformer w/ order(*)	original	**100.0**	**100.0**	**100.0**
	reordered	*99.95*	*99.98*	*99.95*

augmented data significantly improved from 1,055.8 to 5.5. The numbers of deletion/insertion/substitution errors were also reduced, from 583.5 to 2.8 for deletion errors in the MR values, from 1.8 to 0.0 for insertion errors in the MR values, from 109.8 to 2.5 for substitution errors in the MR values, and from 1,842.5 to 4.8 for substitution errors in the MR order. These results suggest the effectiveness of our data augmentation method.

Table 14 shows the perplexity results: while TGEN has a high perplexity of 107.08, JUG and Transformer without order constraints have low values of 81.10 and 80.87, suggesting that they generate relatively fluent sentences. On the other hand, the values for the order-constraints Transformer models are above 90, suggesting that it is slightly less fluent than the methods without order constraints. However, the perplexity of the reference data is as high as 116.91, and it is necessary to confirm this by human evaluation.

The PLL scores are also shown in Table 14. Similar to the results for perplexity, indicating that the scores for TGEN, JUG and Transformer without order constraints are better than those for the order-constraints Transformer models. On the other hand, the score for reference data is low at -88.16, suggesting that the reference data contains sentences that lack naturalness.

Table 13. MRcheck errors of Transformer w/ order models (O: trained using original training data, A: trained using augmented training data) (vd: deletion errors in MR values, vi: insertion error in MR values, vs: substitution error in MR values, os: substitution error in MR order). NEA denotes non-empty attributes.

# NEA	Original test data			Reordered test data		
	number of data	number of errors		number of data	number of errors	
		O	A		O	A
1	0	0	0	0	0.0	0.0
2	1	0	0	0	0.0	0.0
3	92	0	0	4	0.0	0.0
4	311	2	0	546	30.0	1.0
5	631	0	0	2,990	203.5	0.5
6	1,114	0	0	5,570	427.8	2.0
7	1,422	5	0	7,110	814.0	1.0
8	984	12	0	4,920	1,055.8	5.5
Total	4,555	19	0	21,140	2,531.0	10.0
		vd: 2	vd: 0		vd: 583.5	vd: 2.8
		vi: 0	vi: 0		vi: 1.8	vi: 0.0
		vs: 0	vs: 0		vs: 109.8	vs: 2.5
		os: 17	os: 0		os: 1,842.5	os: 4.8

Table 14. Results of perplexity with GPT-2 and Pseudo-log-likelihood (PLL) with BERT: **Bold** indicates best result for original test data (†: order constraints given in decoder, *: trained using augmented training data)

Method	Perplexity(↓)	PLL(↑)
reference	116.91	-88.16
TGEN	107.08	**-74.91**
JUG	81.10	-75.83
Transformer w/o order	**80.87**	-76.61
Transformer w/ order	95.33	-79.48
Transformer w/ order(†)	99.45	-79.23
Transformer w/ order(*)	91.57	-79.20

4.6 Human Evaluation

We also conducted a human evaluation on the MR2T results. Here, the results of the experiments in the previous section confirm that the model with the order constraints input to the encoder performs better than the model with the order constraints input to the decoder, so the latter is not used in this evaluation.

Native English-speaking crowdworkers[10] rated the naturalness, the adequacy [1], and the focus of the generated text packages of 150 selected examples from the original test data. The 150 examples were randomly selected after conditioning the attribute distribution that appeared first in the text package to be approximately uniform, as shown in Table 15. Three workers evaluated each text passage. To avoid any misunderstanding that the values of attributes `name` and `near` are emphasized because they are shown in uppercase, we capitalized only the first letter of each word of those attributes. We gave the instructions shown in Figures 22, 23, and 24 in Appendix B and showed a pair of the MR and the text passages from each of the 150 examples (Figures 25, 26, and 27 in Appendix B) to the evaluators who evaluated them.

Table 15. Number of samples in terms of attributes that first appear in text passage

First attribute	Training data	Validation data	Test data	Selected data for human evaluation
`name`	25,783	2,787	2,727	19
`eatType`	1,608	260	299	19
`food`	2,573	137	197	18
`priceRange`	2,426	165	230	19
`customer rating`	1,389	167	83	18
`area`	2,773	407	363	19
`familyFriendly`	2,000	258	267	19
`near`	2,008	308	389	19
Total	40,560	4,489	4,555	150

[10]We utilized *Prolific* (`https://www.prolific.co/`) and paid each worker thirteen pounds per hour.

Naturalness The evaluators gave scores on a 6-point Likert scale (higher is better) on the following four questions:

1. Is the sentence *natural*?
2. Is the sentence *grammatical*?
3. Is the sentence *comprehensible*?
4. Is the sentence *acceptable* as English, even if it is not natural / grammatical / comprehensible?

Adequacy The evaluators answered the following two questions with either "*yes*" or "*no*":

1. Does the generated sentence meet all the MR values?
2. Does the generated sentence meet all the MR values and the MR orders?

Focus In this experiment, we assumed that the emphasized attribute-value pair appears first in the text passage [85]. The evaluators answered the following question with either "*yes*" or "*no*":

- Is [(`attribute`) value] the focused attribute-pair in the sentence?

Here "[(`attribute`) value]" is the attribute-value pair which appears first in each text passage.

The results are shown in Tables 16 and 17. The naturalness scores of the three baselines are comparable: TGEN, JUG, and Transformer w/o order. On the other hand, the scores for our proposed methods with order constraints and reference are slightly lower than those of the baselines, because it is natural for restaurant recommendation sentences to start with a `name` attribute, and sentences starting with other attributes lose some naturalness. Table 15 shows that the text passages in more than half of the E2E refined dataset start with the `name` attribute. This distribution causes the value of the `name` attribute to always appears first in the generated text passages of the baseline methods. Tables 18 and 19 also list the five examples with the lowest naturalness scores, and the generated text passages

with lower scores in terms of naturalness tend to start with attributes other than `name`. However, looking at the acceptable scores in Table 16, there is no significant difference between the baseline methods and our proposed methods, and although the order control has slightly lost some naturalness, our proposed methods can generate good English without any problems. Table 20 shows an example of the generated text passages for an MR. The reference text's style is rather free, whereas the others resemble templates. This situation might explain why the reference scores are worse than the others.

Table 16. Results for human evaluation in naturalness with 95% confidence interval. **Bold** indicates best result. *: trained using augmented training data.

Method	Naturalness			
	Grammatical	Comprehensible	Natural	Acceptable
Reference	4.12±0.13	4.96±0.10	3.91±0.13	5.15±0.09
TGEN	4.76±0.10	5.24±0.08	**4.86**±0.09	5.35±0.08
JUG	4.78±0.10	**5.28**±0.07	4.77±0.10	**5.36**±0.07
Transformer w/o order	**4.80**±0.10	5.27±0.07	4.86±0.09	**5.36**±0.08
Transformer w/ order	4.44±0.12	5.10±0.09	4.36±0.12	5.26±0.09
Transformer w/ order(*)	4.47±0.12	5.08±0.09	4.36±0.12	5.27±0.09

Table 17. Results for human evaluation in adequacy, and focus. **Bold** indicates best result. *: trained using augmented training data.

Method	Adequacy		Focus
	MR value	MR value & order	
Reference	89.11	88.89	86.22
TGEN	92.22	4.22	56.44
JUG	94.44	4.00	53.78
Transformer w/o order	**95.56**	5.56	55.33
Transformer w/ order	94.67	**92.89**	**91.33**
Transformer w/ order(*)	94.89	92.44	90.67

For adequacy, we found that the generated text passages of the proposed methods appropriately met almost all the MR values and orders. A comparison of the adequacy results with those of MRcheck (Table 12) identifies a clear corre-

lation, suggesting that MRcheck works as an effective automatic measure of the reliability of synthetic data.

For focus, the scores of our proposed methods significantly outperformed those of the baselines. A comparison of the adequacy and focus results shows a clear correlation. This means that correct control of the content order also allows for correct control of the emphasized attribute-value pair in the generated sentences.

4.7 Conclusion

We proposed an MR-to-text method that controls the order of the MR values in generated text passages using MR order constraints. Our proposed method worked effectively and precisely controlled the content order in automatic evaluations. Data augmentation also effectively improved the performance using the MR2T and T2MR models to balance the data distribution in terms of non-empty attributes. The human evaluation results suggest that our proposed methods can focus attribute-value pairs for correct emphasis by controlling the content order. Even though the proposed methods suffered slightly less naturalness compared with the baselines, they generated proper English sentences without any problem. Future work will control such other aspects as the text structure and the output length and apply such methods to other MR-to-text datasets.

Table 18. Examples with five lowest scores for human evaluations (G: Grammatical; C: Comprehensible).

Metrics	Score	Method	Text
G	1.33	TGEN	NAME is a children friendly pub in the city centre near NEAR. It is in the high price range.
	1.33	Reference	5 out of 5 for this pub, although no facilities for children. It is close to NEAR, called NAME near to the riverside has a price list of more than £30 serves Japanese cuisine.
	1.67	Reference	There is a children friendly English restaurant in the riverside area. It is high price range. It is called NAME, and is located near NEAR.
	1.67	Transformer w/ order	Moderately priced NAME is a non kid friendly restaurant located in the city centre, near NEAR. It serves Chinese food.
	1.67	Transformer w/ order	There is a 5 out of 5 pub that is not children friendly near NEAR called NAME in the riverside area. The price range is more than £30 and Japanese food.
C	3.00	Transformer w/ order	Near NEAR is a restaurant that is family friendly in the riverside area called NAME. It serves Italian food and is cheap.
	3.00	Reference	A kids friendly Japanese pub along the riverside is called NAME and is next to NEAR.
	3.33	Reference	Near NEAR their is a restaurant that is family friendly along the riverside named NAME which serves Italian food on a cheap price range.
	3.33	TGEN	NAME is a children friendly pub in the city centre near NEAR. It is in the high price range.
	3.33	Reference	Located near NEAR in the riverside area is an eat type pub called NAME is a children friendly that serves Japanese food and has a price range more than £30.

Table 19. Examples with five lowest scores for human evaluations (N: Natural; A: Acceptable; *: trained using augmented training data).

Metrics	Score	Method	Text
N	1.33	TGEN	NAME is a children friendly pub in the city centre near NEAR. It is in the high price range.
	1.33	Reference	5 out of 5 for this pub, although no facilities for children. It is close to NEAR, called NAME near to the riverside has a price list of more than £30 serves Japanese cuisine.
	1.67	Reference	3 out of 5 star restaurant style restaurant NAME offers child friendly atmosphere near NEAR.
	2.00	Reference	If you are looking for a high quality, family-friendly dining experience in the heart of city centre, NAME is for you. This pub I near NEAR and serves fast-food like it's gourmet.
	2.00	Reference	Near NEAR by riverside is a pub that is yes family friendly with a low customer rating called NAME and the prices are less than £20.
A	3.67	Reference	Located near NEAR in the riverside area is an eat type pub called NAME is a children friendly that serves Japanese food and has a price range more than £30.
	3.67	Transformer w/o order	NAME is a Japanese pub in riverside near NEAR. It is children friendly and has a high customer rating.
	3.67	Transformer w/ order(*)	Moderately priced NAME is a non kid friendly restaurant located in the city centre. It serves Chinese food and is near NEAR.
	3.67	TGEN	NAME is a children friendly Japanese pub in the city centre near NEAR with a high price range and a customer rating of 5 out of 5.
	3.67	Reference	5 out of 5 rated child friendly NAME, serves Japanese food in a pub at higher than normal prices. Located in the city centre near NEAR.

Table 20. One example for an MR(`name`: NAME (order=3), `eatType`: restaurant (order=2), `customer rating`: 3 out of 5 (order=1), `familyFriendly`: yes (order=4), and `near`: NEAR (order=5)) where TGEN and JUG have good scores and reference has bad scores for human evaluation (G: grammatical; C: comprehensible; N: natural; A: acceptable; *: trained using augmented training data).

Method	Text	Score			
		G	C	N	A
Reference	3 out of 5 star restaurant style restaurant NAME offers child friendly atmosphere near NEAR.	1.67	4.67	1.67	5.33
TGEN	NAME is a three star family friendly restaurant located near NEAR.	6.00	5.67	6.00	6.00
JUG	NAME is a family friendly restaurant with a customer rating of 3 out of 5. It is located near NEAR.	6.00	5.67	6.00	6.00
Transformer w/o order	NAME is a family friendly restaurant located near NEAR. It has a customer rating of 3 out of 5.	6.00	5.67	5.67	6.00
Transformer w/ order	There is a 3 star restaurant NAME that is family friendly located near NEAR.	4.00	5.33	4.33	5.67
Transformer w/ order(*)	There is a three star restaurant NAME that is family friendly located near NEAR.	5.00	5.67	4.00	6.00

5. Conclusion

5.1 Conclusion

In this paper, we studied an MR-to-text method that enables content order control. Since this method is based on supervised learning, it requires a dataset that correctly describes the order in which MR data appear in sentences. In Chapter 3, we developed the E2E refined dataset by correcting many errors in the original E2E dataset. We also added additional annotations, such as ordering information. It was confirmed that this error reduction improved the accuracy of sentence generation. However, its refinement took work and effort, as it had to be done manually. Therefore, applying this refinement method to other datasets would take much work.

In Chapter 4, we used the dataset to develop an MR-to-text method that can control the order of the MR values in a generated text passage based on the given order constraints. Here, it was found that the proposed method using Transformer can generate better sentences than the existing method using LSTM. For controlling the order, it was also found that the control can be achieved with very high accuracy by using a model that simultaneously provides the order constraints in addition to data values as input. Furthermore, it was found that more accurate control is possible if the order constraints are passed to the encoder rather than the decoder. It was also found that the issues caused by the imbalance of the dataset described in Chapter 3, in particular the low accuracy of the order control in instances with many non-empty attributes, can be solved by training on automatically augmented data using both the MR2T model, which generates sentences from MRs, and the T2MR model, which infers MRs from sentences. However, subjective evaluation experiments confirmed that sentences generated using order control are acceptable as English but lose their naturalness slightly. Actually, in the same experiment, it was confirmed from the evaluation results of the reference data constructed in a different order from the general one were low in terms of naturalness. Therefore, it seems that the order control method does not cause the loss of naturalness, but further studies are needed to produce more natural sentences.

5.2 Future Works

Data-to-text methods should not always generate one type of text from a single piece of data but should generate different types of text depending on various factors, such as who is reading the generated text, what particular emphasis is placed on the message, what kind of interaction has taken place so far, and so on. This paper has described a method that can freely control the order of input contents with high performance. However, issues still need to be addressed regarding control by factors other than order, the naturalness of the generated sentences, and the variety of the dataset used. In addition, although this method uses a model learned from a single dataset, the use of large language models learned from a large amount of text data, which is becoming mainstream these days, should also be considered.

5.2.1 Control by Factors Other Than Order

Although controlling the content order was the topic of this study, other control elements such as sentence length and number of sentences can also be possible parameters of controllability. As an example, consider a model where, in addition to the order constraints, the number of sentences and the sentence indexes annotated in Section 3.4.2 are given to the encoder as the other constraints. For example, it takes [“⟨*sos*⟩”, “*3*”, “*NAME*”, “*moderate*”, “*restaurant*”, “*Italian*”, “⟨*sep*⟩”, “*city centre*”, “*NEAR*”, “⟨*sep*⟩”, “*yes*”, and “⟨*eos*⟩”] as input to the encoder, when the MR shown in Table 4 is used. That is, the number of sentences (“*3*”) is given immediately after “⟨*sos*⟩” and the special symbol token “⟨*sep*⟩” is inserted into the token whose sentence index changes. The model architecture and the experimental conditions are the same as described in Section 4.4.4 and Section 4.5. As the results listed in Table 21 and 22, controlling by three types of information - order, number of sentences, and sentence indexes (Transformer w/ 3 constraints(♣)) - is more accurate than controlling by order alone (Transformer w/ order).

Here, let us consider another model for length control. Length control and order control are independent. As the annotation of sentence length does not exist in the E2E refined dataset, we defined it as follows:

Table 21. Results of automatic evaluation: **Bold** indicates best result for order-independent reference, and *Italic* indicates best result for order-dependent reference (I: order-independent reference, D: order-dependent reference, †: order constraints given in decoder, *: trained using augmented training data, ♠: two constraints (order and length) given in encoder, ♣: three constraints (order, number of sentences, and sentence indexes) given in encoder)

Method	Ref.	BLEU(↑)	NIST(↑)	METEOR(↑)	ROUGE_L(↑)	CIDEr(↑)
TGEN	I	0.5626	7.8907	0.4278	0.6614	2.4066
	D	0.3339	5.6815	0.3762	0.5262	2.0885
JUG	I	0.5733	7.6896	0.4337	0.6488	2.3972
	D	0.3505	5.6242	0.3871	0.5201	2.0580
Transformer	I	0.5840	7.8227	0.4384	0.6659	2.5141
w/o order	D	0.3600	5.7151	0.3910	0.5344	2.1753
Transformer	I	0.6280	8.6083	0.4595	0.7551	2.7783
w/ order	D	0.4836	7.0947	0.4356	0.7422	3.3062
Transformer	I	0.6232	8.6180	0.4562	0.7489	2.7571
w/ order(†)	D	0.4771	7.1033	0.4318	0.7352	3.2709
Transformer	I	0.6335	8.6198	0.4624	0.7575	2.7855
w/ order(*)	D	0.4914	7.0941	0.4393	0.7453	3.3383
Transformer	I	0.6278	8.6228	0.4600	0.7554	2.8116
w/ 2 constraints(♠)	D	0.4882	7.1265	0.4372	0.7431	3.3948
Transformer	I	**0.6518**	**8.9362**	**0.4660**	**0.7682**	**2.9066**
w/ 3 constraints(♣)	D	*0.5218*	*7.5579*	*0.4452*	*0.7576*	*3.6140*

Table 22. MRcheck results of accuracy: **Bold** indicates best result for original test data, and *Italic* indicates best result for reordered test data, †: order constraints given in decoder, *: trained using augmented training data, ♠: two constraints (order and length) given in encoder, ♣: three constraints (order, number of sentences, and sentence indexes) given in encoder)

Method	Test data	MR value&order	MR value	MR order	Sentence index	# sentences	Length
TGEN	original	6.74	99.41	6.74	n/a	n/a	n/a
JUG	original	5.93	98.24	5.93	n/a	n/a	n/a
Transformer w/o order	original	7.42	100.0	7.42	n/a	n/a	n/a
Transformer w/ order	original	99.58	99.96	99.58	n/a	n/a	n/a
	reordered	88.03	96.74	88.07	n/a	n/a	n/a
Transformer w/ order(†)	original	94.64	99.58	94.67	n/a	n/a	n/a
	reordered	56.10	94.87	56.14	n/a	n/a	n/a
Transformer w/ order(*)	original	**100.0**	**100.0**	**100.0**	n/a	n/a	n/a
	reordered	*99.95*	*99.98*	*99.95*	n/a	n/a	n/a
Transformer w/ 2 constraints(♠)	original	99.50	99.58	99.54	n/a	n/a	95.76
Transformer w/ 3 constraints(♣)	original	98.97	99.65	98.88	98.97	99.65	n/a

- The sentence length is defined as "short", "middle", and "long".
- "long": sentences that satisfy $n_i > (n_{ave}) + (n_{max} - n_{ave})/2$,
- "short": sentences that satisfy $n_i < (n_{ave}) - (n_{ave} - n_{min})/2$,
- "middle": all other sentences,

where n_i indicates the total number of tokens in sentence i, $n_{ave}/n_{max}/n_{min}$ indicate the average/maximum/minimum number of tokens across all sentences in the dataset, respectively. For example, it takes ["$\langle sos \rangle$", "*short*", "*NAME*", "*moderate*", "*restaurant*", "*Italian*", "*city centre*", "*NEAR*", "*yes*", and "$\langle eos \rangle$"] as input to the encoder, when the MR shown in Table 7 is used. That is, the sentence length "*short*" is given immediately after "$\langle sos \rangle$". The model architecture and the experimental conditions are the same as described in Section 4.4.4 and Section 4.5. As the results listed in Table 21 and 22, controlling for two types of information, order and length (Transformer w/ 2 constraints (♠)), produces more accurate sentences than controlling for order alone (Transformer w/ order).

As these examples show, if we would like to control the generation of a sentence with some conditions, we can pass information indicating those conditions to the encoder to control it with high accuracy.

Here, we have only mentioned existing annotations and annotations that can be generated automatically. However, it would be challenging to annotate other control information, such as sentence style, automatically. Therefore, it will be necessary to develop the dataset, considering the validity of the control by that condition and the cost required for the annotation.

5.2.2 Ensuring Naturalness

According to human evaluation in Section 4.6, there was a tendency for the naturalness of sentences generated by reordering to decrease. This is because restaurant introductions are composed in a certain order of contents, and changing the order itself would decrease naturalness. On the other hand, another reason may be that there needs to be more variety of data with various orders in the training data set. Therefore, collecting such data may make learning a model that generates more natural sentences possible. In addition, corpus expansion with the help of the recent large language model should be considered.

5.2.3 Expansion to Various Datasets

In this study, the content order-controllable method was investigated using the E2E refined dataset. Here, let us now consider what considerations would be necessary for other datasets. As this method is trained in a supervised manner possessing the correct labels, the labels of the order information in which each value in the dataset appears in the sentence are essential. Therefore, as described in Chapter 3, manual labelling work at a considerable cost would be required. However, as with the E2E refined dataset, if the data format is an attribute-value pair (MR) and each data value appears only once in a sentence, it can be inferred using the following method.

Consider the T2MR model described in Section 4.4.3, in which only MR values are inferred. Among the experimental conditions described in Section 4.5.2, we changed the number of heads and that of layers to 1, respectively. We also changed the input to the decoder to the fixed position values [“*0*”, “*1*”, . . ., “*7*”] only. In

this case, the values of all eight attributes are obtained in a non-auto-regressive manner. For example, if the input text to this model is "*NAME is a family friendly Chinese pub in the riverside near the NEAR*," then ["*NAME*", "*pub*", "*Chinese*", "", "", "*riverside*", "*yes*", "*NEAR*"] are obtained as a sequence of MR value tokens of length 8. Here, the fourth and fifth tokens are empty strings, indicating that the values for the `priceRange` and `customer rating` attributes are empty.

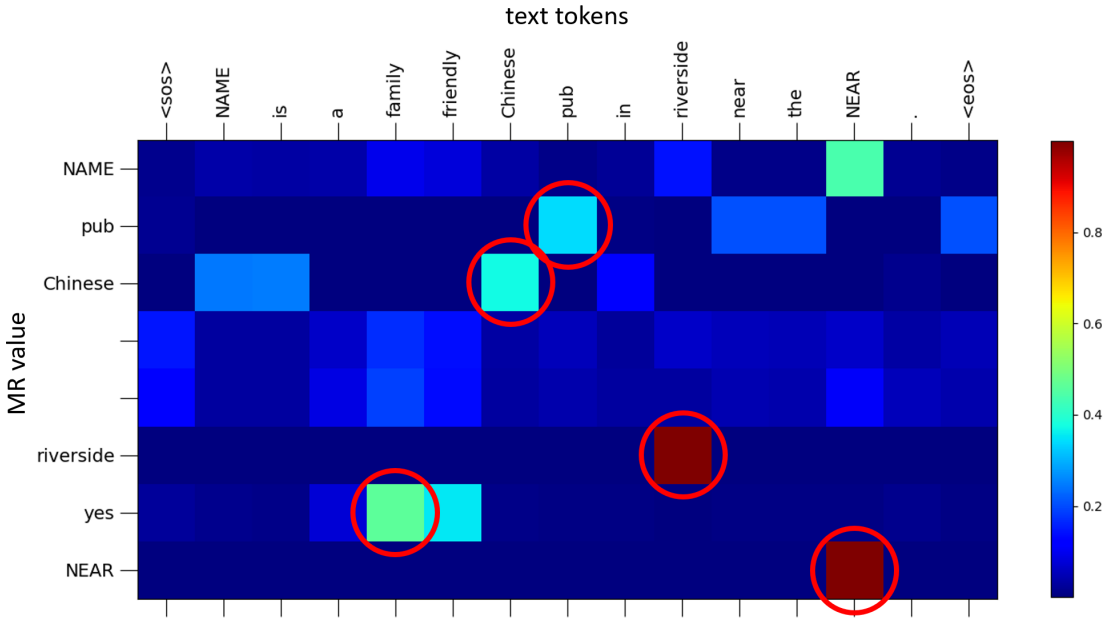

Figure 21. Cross attention weight. The vertical axis indicates MR value tokens and the horizontal axis indicates the tokens in the sentence.

Figure 21 shows the cross-attention weight between the input text tokens and the inferred MR value tokens when the T2MR inference is performed. That is, out of all the tokens in the text, the weight for the token most closely related to the inferred MR value token is learnt to be the largest. Therefore, the order in which the tokens with the highest weights appear can be regarded as the MR order as it is. It should be noted that the "*NAME*" token is always included in the text, so the attention value for the token does not seem to represent the relationship between the text and the MR value well. Therefore, for the `name` attribute, the position of the "*NAME*" token in the text should be considered

instead of using the cross attention weight. As shown in Table 23, the accuracy of the MR order estimation for the test data is not perfect but is as high as 95.96%. This result suggests the potential for automatic estimation of the order information.

Table 23. Accuracy of MR order estimation

dataset	accuracy[%]
training	96.97
validation	95.88
test	95.96
all	96.78

However, for this method to work well, the following two conditions must be fulfilled:

- the labels of the MR values in the dataset must be correctly annotated,
- the character string corresponding to the MR value must appear only once in the text.

Thus, for example, if the dataset includes the following text passage "*Cocco is an Italian restaurant. Cocco is located in the riverside.*", the value "*Cocco*" of the `name` attribute appears twice in the text, it would be difficult to apply this method.

Now consider a dataset provided in resource description frameworks (RDFs) rather than MR, e.g. WebNLG [25]. If the unit of information to be controlled is only a value, assigning an order-of-occurrence label to each value in each triple would be sufficient. For example, if the input data has only one triple such as {`subject`: "*Aarhus Airport*", `predicate`: "*runwayLength*", `object`: "*2702.0*"} and the corresponding text passage is "*Aarhus Airport's runway length is 2702.0.*", it would be possible to control the exact order by assigning ordering information to each subject/predicate/object.

However, if the input data has multiple triples, it would be reasonable to be able to control them with units of triples. In this case, the relationship between triples should be considered, and it is necessary to assign labels not only to values

Table 24. Example of WebNLG dataset. As the order information are not included in the dataset, added by manually.

Shape	(X (X) (X (X) (X) (X)))
Shape type	mixed
Size	5
RDF(1)	predicate: {value: cityServed, order: 2} subject: {value: Abilene Regional Airport, order: 2} object: {value: Abilene, Texas, order: 3}
RDF(2)	predicate: {value: isPartOf, order: 4} subject: {value: Abilene, Texas, order: 3} object: {value: Texas, order: 5}
RDF(3)	predicate: {value: runwayLength, order: 1} subject: {value: Abilene Regional Airport, order: 2} object: {value: 2194.0, order: 1}
RDF(4)	predicate: {value: country, order: 5} subject: {value: Abilene, Texas, order: 3} object: {value: United States, order: 6}
RDF(5)	predicate: {value: isPartOf, order: 3} subject: {value: Abilene, Texas, order: 3} object: {value: Jones County, Texas, order: 4}
Text	With a runway length of 2194.0, Abilene regional airport serves Abiliene, part of Jones Country, Texas, in the United States.

but also to the order of appearance of the triples themselves. For example, in the case of data listed in Table 24, order information should be assigned to the predicate as the order of appearance of the triples themselves. The respective order of appearance information should be assigned for each subject and object. In case that one value is contained across multiple triples (e.g. object in RDF(1), subject in RDF(2), subject in RDF(4) and subject in RDF(5) have the same value "*Abilen, Texas*"), the same order information should be given to all of them.

Note that there is no guarantee that the automatic inference methods described above will work correctly for data sets with such complex data structures, so manual annotation will still likely be required. In addition, delexicalization is only effective when the data values are guaranteed to appear only once in a sentence, so applying it is challenging when such a relationship cannot be guaranteed. Furthermore, as the method studied in Chapter 4 is trained in a supervised manner, if a new dataset is developed, the model must be re-trained using that dataset.

5.2.4 Combination with Large Language Model

In this work, the model of our proposed method was trained only from the training data of the E2E refined dataset. However, many large language models trained on a large amount of text data have been published recently, and they can generate very natural sentences. Therefore, sentence generation using those models is one of the most important issues to be considered. For example, Chen et al. [10], Lee et al. [51], and Chang et al. [8] have studied the combination with GPT-2 [71] to generate more accurate sentences. Kale et al. [41] utilized T5 [72] pre-training outperformed models based on pre-training language models such as GPT-2 and BERT [14]. There must also be a possibility of generating sentences with correct order control by providing appropriate prompts for models such as ChatGPT [94].

The following are conversations with Microsoft Bing [11], which uses GPT-4 [66]. The first utterance to Bing was: *Suppose you are a data-to-text system. If the input data is {name: {value: THE WATERMAN, order: 1}, eatType: {value: pub, order: 2}, food: {value: Italian, order: 5}, priceRange: {value: less than £20, order: 6}, customer rating: {value: null, order: 0}, area: {value: city*

[11] `https://www.bing.com/` (accessed on 14th December, 2023)

centre, order: 3}, *familyFriendly:* {*value: no, order: 7*}, *near:* {*value: RAJA INDIAN CUISINE, order: 4*}}, *the output text should be "THE WATERMAN pub is located in the city centre area near RAJA INDIAN CUISINE. It has Italian food in the £20 or less price range and is not family-friendly." Can you generate sentences from the following input data?* {*name:* {*value: THE WATERMAN, order: 1*}, *eatType:* {*value: restaurant, order: 2*}, *food:* {*value: Italian, order: 5*}, *priceRange:* {*value: less than £20, order: 6*}, *customer rating:* {*value: null, order: 0*}, *area:* {*value: city centre, order: 3*}, *familyFriendly:* {*value: no, order: 7*}, *near:* {*value: RAJA INDIAN CUISINE, order: 4*}}.

Here, this utterance includes instruction on how to generate sentences from a specific input data, then ask Bing to generate sentences from another input data whose data are slightly changed from the first data (the value of `eatType` attribute is changed from "*pub*" to "*restaurant*").

The response from Bing was: *Sure, based on the input data you provided, the generated sentence would be: "THE WATERMAN restaurant is located in the city centre area near RAJA INDIAN CUISINE. It serves Italian food in the £20 or less price range and is not family-friendly."* The generated sentences meet all the input MR data perfectly.

Then, the second utterance was: *Great! In which case, can you generate another sentences from the following input data?* {*name:* {*value: THE WATERMAN, order: 1*}, *eatType:* {*value: pub, order: 3*}, *food:* {*value: Italian, order: 4*}, *priceRange:* {*value: less than £20, order: 5*}, *customer rating:* {*value: null, order: 0*}, *area:* {*value: city centre, order: 7*}, *familyFriendly:* {*value: no, order: 2*}, *near:* {*value: RAJA INDIAN CUISINE, order: 6*}}. Here, the order constraints are changed, whereas the values are unchanged.

The response was: *Sure, based on the input data you provided, the generated sentence would be: "THE WATERMAN is not family-friendly. It is a pub that serves Italian food in the £20 or less price range. It is located near RAJA INDIAN CUISINE in the city centre area."* The generated sentences in this response also perfectly reflect all the input MR data.

The third utterance was: *Wow! How about this? Can you generate sentences from the following input data?* {*name:* {*value: THE WATERMAN, order: 3*}, *eatType:* {*value: restaurant, order: 2*}, *food:* {*value: null, order: 0*}, *priceRange:*

{*value: null, order: 0*}, *customer rating:* {*value: 3 out of 5, order: 1*}, *area:* {*value: null, order: 0*}, *familyFriendly:* {*value: yes, order: 4*}, *near:* {*value: RAJA INDIAN CUISINE, order: 5*}}. Here, the different MR data are given as input.

The response was: *Sure, based on the input data you provided, the generated sentence would be: "The restaurant has a customer rating of 3 out of 5. THE WATERMAN is family-friendly and is located near RAJA INDIAN CUISINE."* Unfortunately, the order of `eatType` and `customer rating` is permuted in the generated sentences.

Thus, the following sentences were given as the fourth utterance: *Almost! But you made a mistake in the order among "name", "eatType", and "customer rating". Can you try again by following the given order carefully?*

The response to the utterance was: *I apologize for the mistake. Based on the corrected order of the input data, the generated sentence would be: "The restaurant, THE WATERMAN, has a customer rating of 3 out of 5. It is family-friendly and is located near RAJA INDIAN CUISINE."* Unfortunately, the generated sentences in this response did not meet the order constraints either.

As these example conversations show, it can be possible for a pre-trained large language model to generate correct sentences from the given input without fine-tuning, whereas it sometimes makes mistakes. Although it is difficult to deny that the original E2E dataset and our E2E refined dataset were used to train the GPT-4 model, it must be said that the performance is astonishing. The performance of such large language models has been improving at an alarming rate recently, and it will not be long before perfect sentence generation can be achieved without fine-tuning.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Satoshi Nakamura. Since his guidance during my first master's course, he has taught me many things, not only in research but also as a mentor in life. He also gave me a nudge from behind when I made various decisions about this doctoral course, from deciding on the theme to submitting my journal paper. In particular, I wouldn't be taking on the challenge of obtaining a degree if it weren't for that one word he said to me when we met in Tokyo in December 2018. I am genuinely grateful to have been given this opportunity.

I would like to express my sincere gratitude to my co-supervisor, Professor Taro Watanabe. At the interim presentations and public hearings, I received much guidance from different perspectives that I needed to have. I learnt the importance of taking a broad view of things. I will continue to have an overall and holistic perspective in the future.

I would also like to express my genuine gratitude to the other co-supervisor, Associate Professor Katsuhito Sudoh. He gave me detailed guidance with his deep and wide-ranging insight on everything from the various questions that arose during my research to the precise writing of our papers. He also handled various chores, from setting up meetings to instructing me on the detailed rules for carrying out research as a student in the Augmented Human Communication Laboratory.

I would like to thank Matsuda-san and the other members of the Augmented Human Communication Laboratory. Even though we rarely met in the laboratory during my doctoral course, I will never forget the warm welcome I received as a colleague in the same laboratory.

I also thank all the staff at NAIST for their dedicated support in making my student life as a working student living in Tokyo smooth and comfortable.

I am grateful to my colleagues at Sony Group Corporation and Sony Research Inc. for their understanding and consideration of pursuing my doctoral course independently of my work.

Finally, I express my utmost gratitude to my friends and family, who have supported me in my two-footed pursuit of office work and studenthood.

Publication List

Journal Paper (peer-reviewed)

[1] Keisuke Toyama, Katsuhito Sudoh, and Satoshi Nakamura, "Content Order-Controllable MR-to-Text," IEEE Access, Volume 11, pp. 129353–129365, 2023.

International Conference Paper (peer-reviewed)

[1] Keisuke Toyama, Katsuhito Sudoh, and Satoshi Nakamura, "E2E Refined Dataset," in Proceedings of the 26th International Conference on Oriental-COCOSDA, 2023.

References

[1] Jacopo Amidei, Paul Piwek, and Alistair Willis. The use of rating and likert scales in natural language generation human evaluation tasks: A review and some recommendations. In *Proceedings of the 12th International Conference on Natural Language Generation*, 2019.

[2] Peter Anderson, Basura Fenando, Mark Johnson, and Stephen Gould. Guided open vocabulary impage captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*, 2015.

[4] Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. Constrained decoding for neural nlg from compositional representations in task-oriented dialogue. In *Proceesdings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[5] H. Banaee, M. U. Ahmed, and A. Loutfi. Towards nlg for physiological data monitoring with body area networks. In *Proceedings of ENLG*, 2013.

[6] G. Carenini and J. D. Moore. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170, 2006.

[7] Junyi Chai, Reid Pryzant, Victor Ye Dong, Konstantin Golobokov, Chenguang Zhu, and Yu Liu. Fast: Improving controllability for text generation with feedback aware self-training. Technical report, arxiv:2210.03167, 2022.

[8] Ernie Chang. Jointly improving language understanding and generation with quality-weighted weak supervision of automatic labeling. In *Proceedings of EACL 2021*, 2021.

[9] D. L. Chen and R. J. Mooney. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of ICML*, 2008.

[10] Zhiyu Chen. Few-shot nlg with pre-trained language model. In *ACL2020*, 2020.

[11] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[12] J. Clarke and M. Lapata. Discourse constraints for document compression. *Computational Linguistics*, 36, 2010.

[13] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the 9th Workshop on Statistical Machine Translation*, 2014.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 17th Annual Conference of the North AMerican Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[15] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2nd International Conference of Human Language Technology Research*, 2002.

[16] Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. A survey of natural language generation. *ACM Computing Surveys*, 2022.

[17] Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[18] Ondrej Dušek, David M. Howcroft, and Verena Rieser. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, 2019.

[19] Ondrej Dušek and Filia Jurčíček. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54 the Annucal Meeting of the Assocication for Computational Linguistics*, 2016.

[20] Ondrej Dušek, Jekaterina Novikova, and Verena Rieser. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech and Language*, 2020.

[21] Ondřej Dušek, Jekaterina Novikova, , and Verena Rieser. Finding of the e2e nlg challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, 2018.

[22] Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 2022.

[23] Henry Elder, Jennifer Foster, James Barryy, and Alexander O'Connor. Designing a symbolic intermediate representation for neural surface realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation (NeuralGen)*, 2019.

[24] Thiago Castro Ferreira, Helena Vaz, Brian Davis, and Adriana pagano. Enriching the e2e dataset. In *Proceedings of the 14th International Conference on Natural Language Generation*, 2021.

[25] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating training corpora for nlg micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[26] A. Gatt, F. Porter, E. Reiter, J. R. Hunter, S. Mahamood, W. Moncur, and S. Sripada. From data to text in the neonatal intensive care unit: using nlg technology for decision support and information management. *AI COmmunications*, 22, 2009.

[27] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61, 2018.

[28] Sebastian Gehrmann, Falcon Z. Dai, and Henry Elder. End-to-end content and plan selection for data-to-text generation. In *Proceedings of The 11th International Natural Language Generation Conference*, 2018.

[29] E. Goldberg, N. Driedger, and R. I. Kittredge. Using natural-language processing to produce weather foreacats. *IEEE Expert*, 9, 1994.

[30] Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[31] Qipeng Guo, Zhijing Jin, Ning Dai, Xipeng Qiu, Xiangyang Xue, David Wipf, and Zheng Zhang. P2: A plan-and pretrain approach for knwledge graph-to-text generation. In *Proceedings of 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, 2020.

[32] Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *CoRR*, abs/2006.04702, 2020.

[33] M. D. Harris. Building a large-scale commercial nlg system for an emr. In *Proceedings of INLG*, 2008.

[34] Sepp Hochereiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9, 1997.

[35] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[36] Xinyu Hua and Lu Wang. Sequence-level content planning and style specification for neural text generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

[37] D. Huske-Kraus. Suregen-2: a shell system for the generation of clinical documents. In *Proceedings of EACL*, 2003.

[38] W. J. Hutchines and H. L. Somers. An introduction to machine translation. *Academic Press London*, 362, 1992.

[39] Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. Learning to select, track, and generate for data-to-text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[40] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48, 2022.

[41] Mihir Kale and Abhinav Rastogi. Text-to-text pre-training for data-to-text tasks. In *Proceedings of The 13th International Conference on Natural Language Generation*, 2020.

[42] Katharina Kann, Sascha Rothe, and Katja Filippova. Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd CoNLL*, 2018.

[43] Zdenek Kasner and Ondrej Dusek. Neural pipeline for zero-shot data-to-text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguisitics*, 2022.

[44] Seiya Kawano, Koichiro Yoshino, and Satoshi Nakamura. Controlled neural response generation by given dialogue acts based on label-aware adversarial learning. *Transactions of the Japanese Society for Artificial Intelligence*, 2021.

[45] Chris Kedzie and Kathleen McKeown. A good sample is hard to find: Noise injection sampling and self-training for neural language generation models. In *Proceedings of the 12th International Conference on Natural Language Generation*, 2019.

[46] Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason D. Williams, and Matthew Henderson. The fourth dialog state tracking challenge. In *Proceedings of the 7th International Workshop on Spoken Dialogue System*, 2016.

[47] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[48] Karen Kukich. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, 1983.

[49] I. Langkilde-Geary and K. Knight. Halogen statistical sentence generator. In *Proceedings of ACL*, 2002.

[50] Remi Lebret, David Grangier, and Michael Auli. 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*, 2016.

[51] Joosung Lee. Transforming multi-conditioned generation from meaning representation. In *Proceedings of RANLP 2021*, 2021.

[52] Yuanmin Leng, François Portet, Cyril Labbé, and Raheel Qader. Controllable neural natural language generation: comparison of state-of-the-art control strategies. In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web*, 2020.

[53] L. Lepp, M. Munezero, M. Granroth-wilding, and H. Toivonen. Data-driven news generation for automated journalism. In *Proceedings of INLG*, 2017.

[54] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004.

[55] Xuming Lin, Shaobo Cui, Zhongzhou Zhao, Wei Zhou, Ji Zhang, and Haiqing Chen. Ggp: A graph-based grouping planning for explicit control of long text generation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021.

[56] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi CHen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arxiv:1907.11692*, 2019.

[57] Yutao Luo, Menghua Lu, Gongshen Liu, and Shilin Wang. Few-shot table-to-text generation with prefix-controlled generator. In *Proceedings of the 29th International Conference on Computational Linguistics*, 2022.

[58] S. W. McRoy, S. Channarukul, and S. S. Ali. An augmented template-based approach to text realization. *Natural Language Engineering*, 9, 2003.

[59] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of NAACL-HLT 2016*, 2016.

[60] Xinhao Mei, Xubo Liu, Mark D. Plumbley, and Wenwu Wang. Automated audio captioning: an overview of recent progress and new challenges. *EURASIP Journal on Audio, Speech, and Music Processing*, 2022.

[61] M. Molina, A. Stent, and E. Parodi. Generating automated news to explain the meaning of sensor data. In *Proceedings of IDA*, 2011.

[62] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[63] Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annucal SIGdial Meeting on Discourse and Dialogue*, 2017.

[64] F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29, 2003.

[65] M. O'Donnell. Ilex: an architecture for a dynamic hypertext generation system. *Natural Language Engineering*, 7, 2001.

[66] OpenAI. Gpt-4 technical report. *arxiv:2303.08774*, 2023.

[67] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[68] F. Portet, E. Reiter, A. Gatt, J. R. Hunter, S. Sripada, Y. Freer, and C. Sykes. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173, 2009.

[69] Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of NAACL-HLT 2018*, 2018.

[70] Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *Proceedings of The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[71] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog (download at https://insightcivic.s3.us-east-1.amazonaws.com/language-models.pdf)*, 2019.

[72] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning witha unified text-to-text transformer. *The Journal of Machine Learning Research*, 21, 2020.

[73] E. Reiter, C. Mellish, and J. Levine. Automatic generation of technical documentation. *Artificial Intelligence*, 9, 1995.

[74] E. Reiter, R. RObertson, and L. M. Osman. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144, 2003.

[75] Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[76] Yuichi Sasazawa. Controlling keywords and their positions in text generation. In *Proceedings of the 16th International Natural Language Generation Conference*, 2023.

[77] Lei Sha, Lili Mou uand Tianyu Liu, Pascal Pupart, Sujian Li, Baobao Chang, and Zhifang Sui. Order-planning neural text generation from structured data. In *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[78] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of NAACL 2018*, 2018.

[79] A. Siddharthan, M. Green, K. van Deemter, C. Mellish, and R. van der Wal. Blogging birds: Generating narratives about reintroduced species to promote public engagement. In *Proceedings of INLG*, 2013.

[80] Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Guiseppe Fiameni, and Rita Cucchiara. From show to tell: A survey on deep learning-based image captioning. *arxiv:2107.06912*, 2021.

[81] O. Stock, M. Zancanaro, P. Busetta, C. Callaway, A. Cruger, M. Kruppa, T. Kuflik, E. Not, and C. Rocchi. Adaptive, intelligent presentation of information for the museum visitor in peach. *User Modeling and User-Adapted Interaction*, 17, 2007.

[82] Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021.

[83] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In *Proceedings of ICML*, 2011.

[84] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing System*, 2014.

[85] Naoaki Suzuki and Satoshi Nakamura. Representing 'how you say' with 'what you say':english corpus of focused speech and text reflecting corresponding implications. In *Proceedings of the 23rd Interspeech Conference*, 2022.

[86] M. Theune, E. Klabbers, J. R. E. de Pijper, E. Krahmer, and J. Odijk. From data to speech: a general approach. *Natural Language Engineering*, 7, 2001.

[87] Keisuke Toyama, Katsuhito Sudoh, and Satoshi Nakamura. Content order-controllable mr-to-text. *IEEE Access*, 11, 2023.

[88] Keisuke Toyama, Katsuhito Sudoh, and Satoshi Nakamura. E2e refined dataset. In *Proceedings of the 26th International Conference on Oriental COCOSDA*, 2023.

[89] Bayu Disiawan Trisedya, Xiaojie Wang, Jianzhong Qi, Rui Zhang, and Qingjun Cui. Grouped-attention for content-seletion and content-plan generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021.

[90] Bo-Hsiang Tseng, Jianpeng Cheng, Yimai Fang, and David Vandyke. A generative model for joint natural language understanding and generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[91] R. van der Wal, N. Sharma, C. Mellish, A. Robinson, and A. Siddharthan. The role of automated feedback in training and retaining biological recorders for citizen science. *Conservation Biology*, 30, 2016.

[92] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems 2017*, 2017.

[93] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of 2015 IEEE conference on Computer Vision and Pattern Recognition*, 2015.

[94] Yuntao Wang, Yanghe Pan, Miao Yan, Zhou Su, and Tom H. Luan. A survey on chatgpt: Ai-generated contents, challenges, and solutions. *arxiv:2305.18339*, 2023.

[95] L. Wanner, H. Bosch, N. Bouayad-Agha, and G. Casamayor. Getting the environmental information across: from the web to the user. *Expert Systems*, 32, 2015.

[96] Sam Wiseman, Stuart Shieber, and Alexander Rush. Learning neural templates for text genration. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[97] Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. Challenges in data-to-document genration. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[98] Xinnuo Xu. Agggen: Ordering and aggregating while generating. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

[99] Takeshi Yamazaki. Procedual text generation from a flow graph. *IPSJ Journal*, 57, 2015.

[100] Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. A survey of deep learning techniques for neural machine translation. *arxiv:2002.07526*, 2020.

[101] Ruslan Yemakov, Nicholas Drago, and Angelo Ziletti. Biomedical data-to-text generation via fine-tuning transformers. In *Proceedings of the 14th International Conference on Natural Language Generation (INLG)*, 2021.

[102] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *arxiv:2303.18223*, 2023.

Appendix

A. Examples

Some examples of the experiments in Section 4.5 are shown in Tables 25, 26, and 27. Comparing Tables 25 and 26, the Transformer models with order constraints accurately reflected the MR values and the MR order in the generated text passages, although TGEN, JUG, and the Transformer models without order constraints did not. Comparing Tables 25 and 27, the Transformer models with order constraints properly preserved the MR order in the generated text passages, although TGEN, JUG, and the Transformer models without order constraints did not.

B. Instructions for Human Evaluation

The instructions for those participating in the human evaluation described in Section 4.6 are shown in Figure 22. The instructions for adequacy and focus are shown in Figures 23 and 24. Screenshots of the human evaluation are shown in Figures 25, 26, and 27. These systems were designed using Google Forms.

Table 25. One example of MR value, MR order, and generated text passages. (*: trained using augmented training data)

		Attribute	Value	Order
MR		`name`	NAME (THE WATERMAN)	1
		`eatType`	pub	2
		`food`	Italian	5
		`priceRange`	less than £20	6
		`customer rating`	(empty)	0
		`area`	city centre	3
		`familyFriendly`	no	7
		`near`	NEAR (RAJA INDIAN CUISINE)	4
Text	Reference	NAME pub is located in the city centre area near NEAR. It has Italian food in the £20 or less price range and is not family-friendly.		
	TGEN	NAME is an Italian pub located in the city centre near NEAR. It is not family-friendly and has a price range of less than £20.		
	JUG	NAME is a pub in the city centre near NEAR. It serves Italian food for less than £20. It is not family-friendly.		
	Transformer w/o order	NAME is a pub that serves Italian food. It is located in the city centre near NEAR. It is not family-friendly and has a price range of less than £20.		
	Transformer w/ order	NAME is a pub in the city centre near NEAR. It serves Italian food for less than £20 and is not family-friendly.		
	Transformer w/ order(*)	NAME is a pub in the city centre near NEAR. It serves Italian food for less than £20 and is not family-friendly.		

Table 26. Another example of generated text passages: MR value is identical to Table 25, although MR order is different. (*: trained using augmented training data)

		Attribute	Value	Order
MR		`name`	NAME (THE WATERMAN)	1
		`eatType`	pub	3
		`food`	Italian	4
		`priceRange`	less than £20	5
		`customer rating`	(empty)	0
		`area`	city centre	7
		`familyFriendly`	no	2
		`near`	NEAR (RAJA INDIAN CUISINE)	6
Text	Reference	NAME is a non family-friendly pub that serves Italian food for less than £20. It is located near NEAR in the city centre area.		
	TGEN	NAME is an Italian pub located in the city centre near NEAR. It is not family-friendly and has a price range of less than £20.		
	JUG	NAME is a pub that serves Italian food for less than £20. It is located in the city centre near NEAR and is not family-friendly.		
	Transformer w/o order	NAME is a pub that serves Italian food. It is located in the city centre near NEAR. It is not family-friendly and has a price range of less than £20.		
	Transformer w/ order	NAME is a non family-friendly pub serving Italian food for less than £20 near NEAR in the city centre.		
	Transformer w/ order(*)	NAME is a non family-friendly pub that serves Italian food for less than £20. It is located near NEAR in the city centre.		

Table 27. Another example of generated text passages: Except the MR value of `eatType`, all MR values and MR orders are identical to Table 25. (*: trained using augmented training data)

		Attribute	Value	Order
MR		`name`	NAME (THE WATERMAN)	1
		`eatType`	restaurant	2
		`food`	Italian	5
		`priceRange`	less than £20	6
		`customer rating`	(empty)	0
		`area`	city centre	3
		`familyFriendly`	no	7
		`near`	NEAR (RAJA INDIAN CUISINE)	4
Text	Reference	NAME restaurant is located in the city centre area near NEAR. It has Italian food in the £20 or less price range and is not family-friendly.		
	TGEN	NAME is a non family-friendly Italian restaurant in the city centre near NEAR with a price range of less than £20.		
	JUG	NAME is a restaurant that serves Italian food for less than £20. It is located in the city centre near NEAR and is not family-friendly.		
	Transformer w/o order	NAME is a non family-friendly Italian restaurant located in the city centre near NEAR. It has a price range of less than £20.		
	Transformer w/ order	NAME is a restaurant in the city centre near NEAR. It serves Italian food for less than £20 and is not family-friendly.		
	Transformer w/ order(*)	NAME is a restaurant located in the city centre near NEAR. It serves Italian food for less than £20. It is not family-friendly.		

An AI system generates sentences from MR (meaning representation) data. Each MR data has up to 8 attribute-value pairs.

```
[MR data]
[1] (name) Blue Spice
[2] (eatType) pub
[3] (food) English
[4] (area) riverside
[5] (familyFriendly) no
[6] (near) Rainbow Vegetarian Cafe

[sentence]
(A) There is a riverside pub near the Rainbow Vegetarian Cafe called Blue Spice. It serves English food but is not family-friendly.
(B) In the riverside area is a pub near Rainbow Vegetarian Cafe called Blue Spice. It serves English food and is not family-friendly.
(C) In riverside, there is a pub near Rainbow Vegetarian Cafe called Blue spice that serves English food. It is not family-friendly.
(D) Blue Spice is an English pub near Rainbow Vegetarian Cafe in the riverside area. It is not family-friendly.
(E) Blue Spice is a pub that serves English food. It is located in riverside near Rainbow Vegetarian Cafe and is not family-friendly.
```

In this case, the MR data has six attribute-value pairs, "[1] (name) Blue Spice", "[2] (eatType) pub", "[3] (food) English", "[4] (area) riverside", "[5] (familyfriendly) no", and "[6] (near) Rainbow Vegetarian Cafe". The AI system can generate sentences by meeting all attribute-value pairs. The system can generate not only one sentence but 2~6 sentences with different styles, as shown above.
In this study, all variations of values for each attribute are listed below:

```
(name) The Cricketers, Giraffe, Blue Spice, etc.
(eatType) coffee shop, pub, restaurant
(food) Chinese, English, French, Indian, Italian, Japanese, fast food
(priceRange) cheap, expensive, less than £20, moderate, more than £30, £20-25
(customer rating) 1 out of 5, 3 out of 5, 5 out of 5, average, high, low
(area) city centre, riverside
(familyFriendly) no, yes
(near) All Bar One, Crowne Plaza Hotel, Raja Indian Cuisine, etc.
```

Sometimes, the values appear in the sentence with different expressions.

```
[MR data]
[1] (name) The Wrestlers
[2] (eatType) restaurant
[3] (food) Japanese
[4] (priceRange) expensive
[5] (area) riverside
[6] (familyFriendly) yes
[7] (near) Raja Indian Cuisine

[sentence]
There is a high price range Japanese restaurant in riverside near Raja Indian Cuisine that is child friendly called The Wrestlers.
```

In this case, the value "expensive" of the (priceRange) attribute does not appear in the sentence. However, "high price" means expensive, so we can say that the generated sentence meets the attribute-value pair of "(priceRange) expensive".

Some examples are listed below:

```
(food) Italian: pasta, spaghetti, etc.
(priceRange) cheap: inexpensive, low price, etc.
(priceRange) moderate: average price, moderately priced, etc.
(priceRange) expensive: high price, upper range price, etc.
(customer rating) 1 out of 5: one star, one to five, 1 star, etc.
(familyFriendly) yes: family friendly, child friendly, children friendly, kid friendly, etc.
(familyFriendly) no: not family friendly, no child friendly, non-children friendly, not kid friendly, etc.
```

Figure 22. Instructions for Human Evaluation

Question (A) asks if the generated sentence meets all the attribute-value pairs.

```
[MR data]
[1] (near) The Cricketers
[2] (area) city centre
[3] (eatType) pub
[4] (name) The Mill
[5] (familyFriendly) no
[6] (food) fast food
[7] (priceRange) moderate

[sentence]
Near The Cricketers in city centre is the pub The Mill. This adult establishment serves fast food at average prices.
```

In this case, the sentence meets all seven attribute-value pairs. Here, the value "moderate" of the (priceRange) attribute is replaced by "average prices" in the sentence. However, in this case, "moderate" and "average" are the same in meaning. Furthermore, the value "no" of the (familyFriendly) attribute is expressed as "adult establishment". Thus, "Yes" should be chosen for the Question (A).

```
[MR data]
[1] (near) All Bar One
[2] (name) Clowns
[3] (eatType) pub
[4] (customer rating) 3 out of 5
[5] (food) Chinese

[sentence]
Clowns is a coffee shop near All Bar One. It is rated 3 out of 5 and is not family-friendly.
```

In this case, there are three errors.

- The value "Chinese" of the (food) attribute does not appear in the sentence.
- The value "pub" of the (eatType) attribute is changed to "coffee shop" in the sentence.
- The MR has no value for the (familyFriendly) attribute, but the phrase "is not family-friendly" is appeared in the sentence. Thus, "No" should be chosen for the Question (A).

Question (B) asks if the generated sentence meets all the attribute-value pairs in collect order.

```
[MR data]
[1] (eatType) coffee shop
[2] (area) city centre
[3] (name) Blue Spice

[sentence]
A coffee shop in the city centre area called Blue Spice.
```

In this case, the sentence meets all three attribute-value pairs. Also, the order of the attribute-value in the generated sentence is "[1] (eatType) coffee shop" -> "[2] (area) ceity centre" -> "[3] (name) Blue Spice", so the sentence meets the order of the attribute-value pairs in the MR data. Thus, "Yes" should be chosen for the Question (B).

```
[MR data]
[1] (name) Blue Spice
[2] (eatType) restaurant
[3] (food) Chinese
[4] (area) riverside
[5] (near) The Valuts

[sentence]
Blue Spice is a Chinese restaurant in riverside near The Vaults.
```

In this case, the sentence meets all five attribute-value pairs. However, the order of the attribute-value in the generated sentence is "[1] (name) Blue Spice" -> "[3] (food) Chinese" -> "[2] (eatType) restaurant" -> "[4] (area) riverside" -> "[5] (near) The Vaults". Hence, the sentence does not meet the order of the attribute-value pairs in the MR data. Thus, "No" should be chosen for the Question (B).

Figure 23. Instructions for Human Evaluation for Adequacy

Suppose you are looking for a restaurant/coffee shop/pub and ask an AI system to recommend a restaurant/coffee shop/pub for you. The AI system extracts some attribute-value pairs from your question, then generates a response according to the extracted attribute-value pairs.

```
[Your question]
I'm in the city centre now. Can you tell me any Chinese restaurants around here?

[MR data]
[1] (area) city centre
[2] (food) Chinese
[3] (eatType) restaurant

[sentence]
In the city centre, there is a Chinese restaurant called The Wrestler.
```

The AI system extracts three attribute-value pairs from your question, then generates the response. In this case, the system searched the restaurant "The Wreslter" from its database, so the generated sentence includes the name.

According to the question, the system supposes that it is important for the restaurant to be placed in the city centre. So, the AI system would emphasize the attribute-value pair "(area) city centre" in the generated sentence. The emphasized attribute-value pair is named "focused attribute-value pair" in the subsequent studies.

Figure 24. Instructions for Human Evaluation for Focus

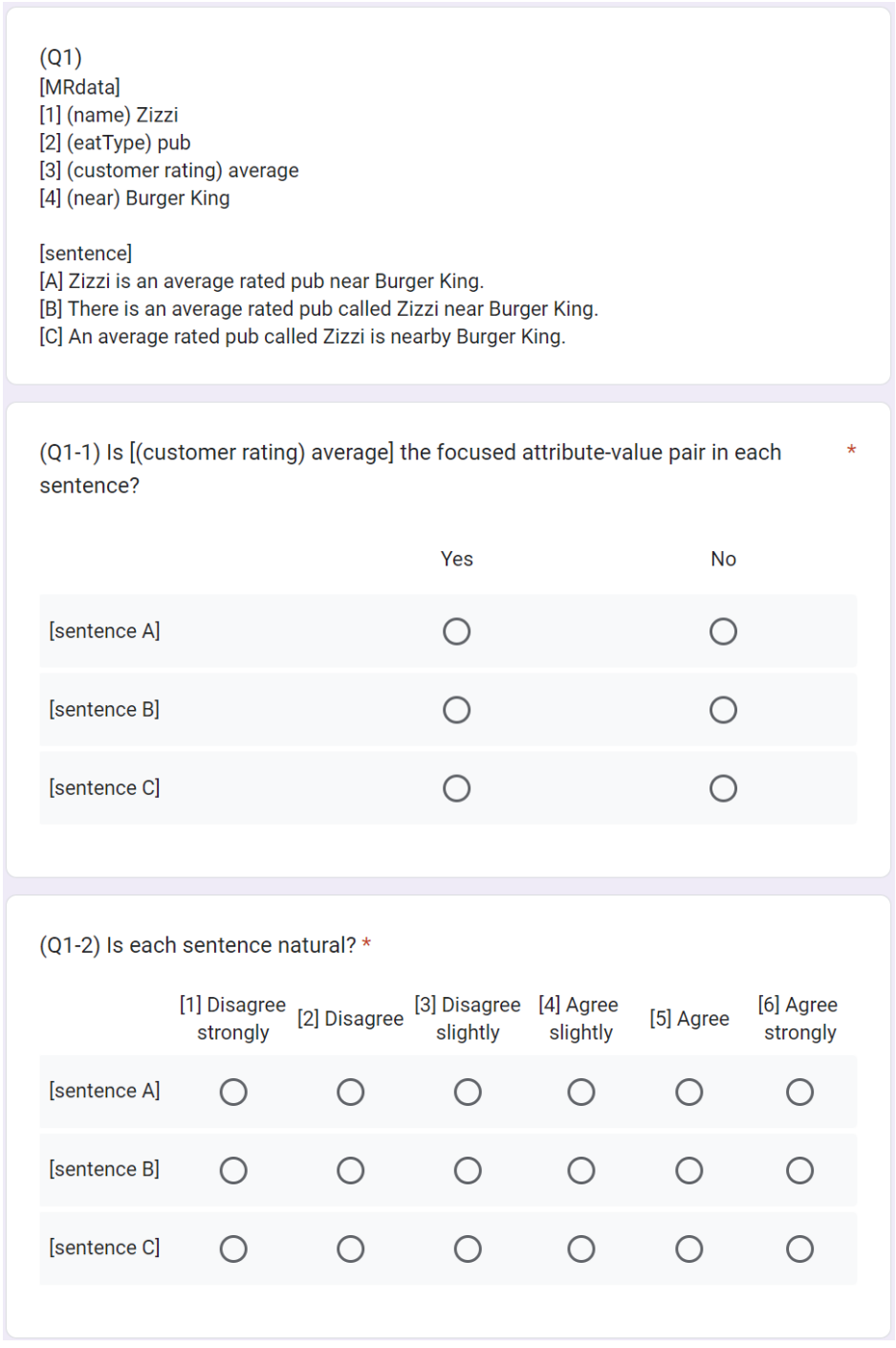

Figure 25. Screenshot of Human Evaluation for Naturalness and Focus (1)

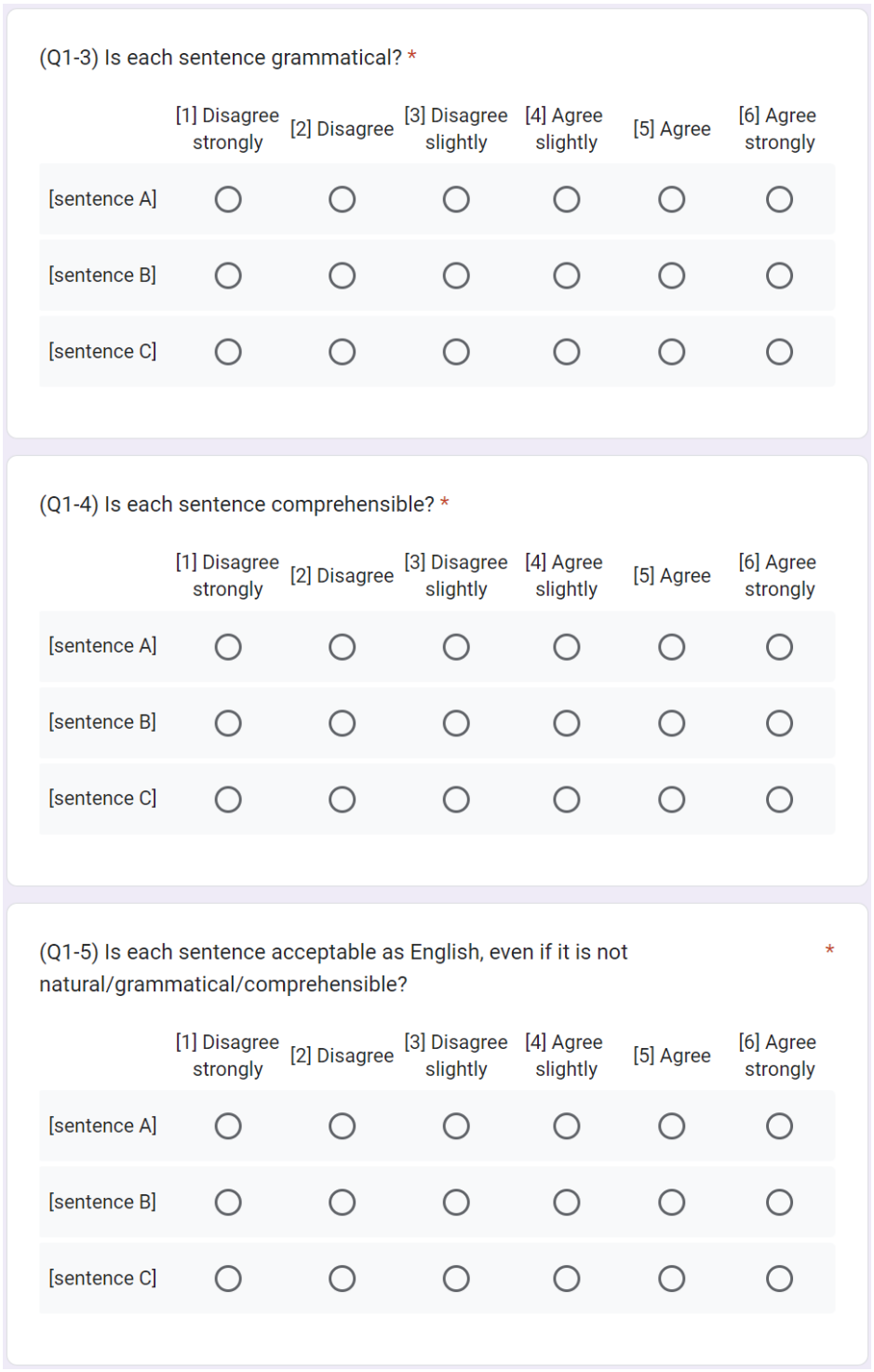

Figure 26. Screenshot of Human Evaluation for Naturalness and Focus (2)

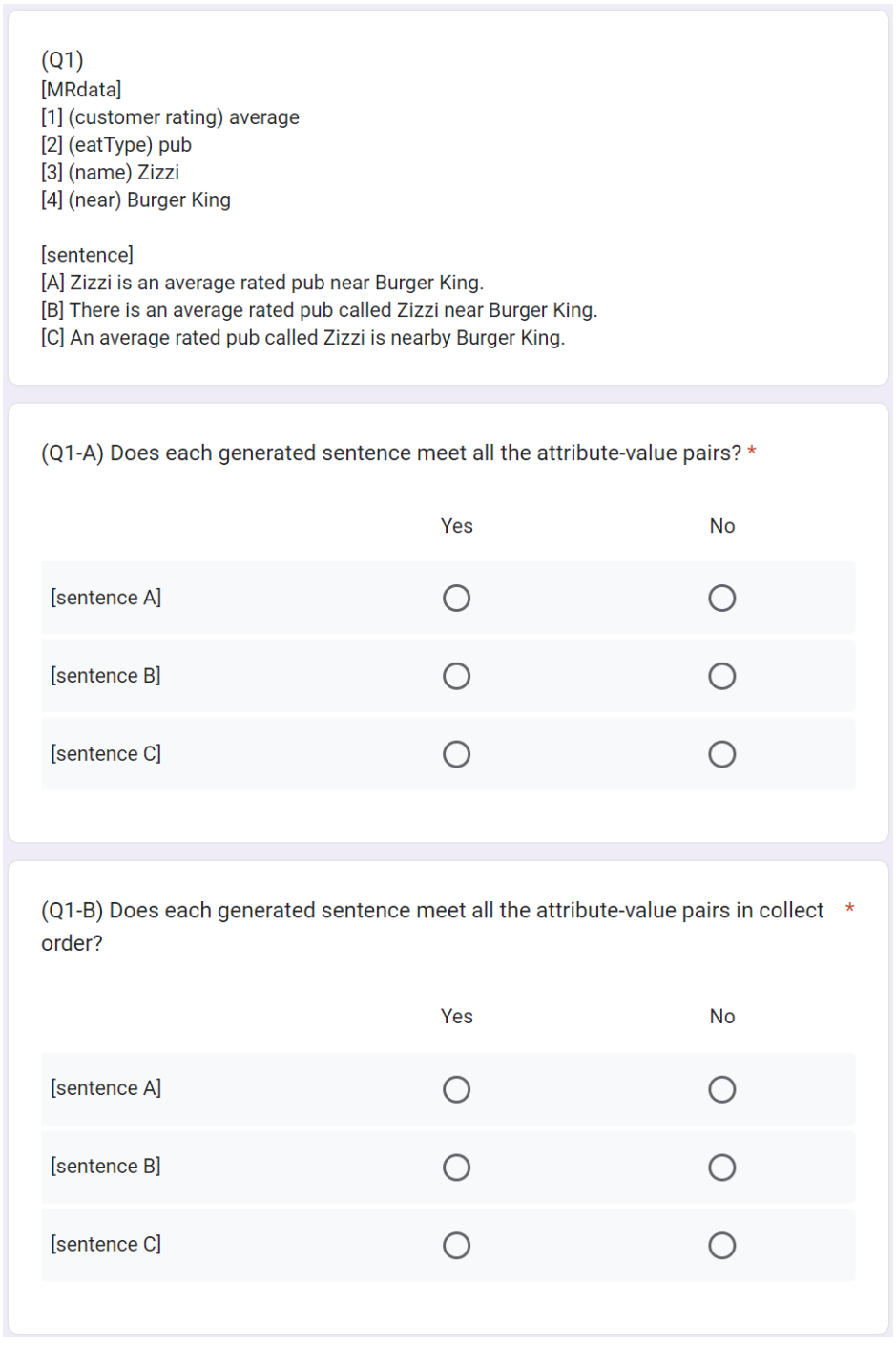

Figure 27. Screenshot of Human Evaluation for Adequacy