Transforming Multi-Conditioned Generation from Meaning Representation

Joosung Lee
Kakao Enterprise Corp., South Korea
rung.joo@kakaoenterprise.com

Abstract

Our study focuses on language generation by considering various information representing the meaning of utterances as multiple conditions of generation. Generating an utterance from a Meaning representation (MR) usually passes two steps: sentence planning and surface realization. However, we propose a simple one-stage framework to generate utterances directly from MR. Our model is based on GPT2 and generates utterances with flat conditions on slot and value pairs, which does not need to determine the structure of the sentence. We evaluate several systems in the E2E dataset with 6 automatic metrics. Our system is a simple method, but it demonstrates comparable performance to previous systems in automated metrics. In addition, using only 10% of the dataset without any other techniques, our model achieves comparable performance, and shows the possibility of performing zero-shot generation and expanding to other datasets.

1 Introduction

In many conversation systems, generating sentences with specific information is useful. For example, it can be used in chatbot systems or spoken dialogue systems to generate utterances that contain *meaning representations* (MRs) corresponding to a user's query. In order to train the NLG system that reflects this variety of information, a large amount of labeled data is required. At the 2017 E2E challenge (Dušek et al., 2019), a large dataset was released, which consisted of pairs of MRs representing restaurant reviews and corresponding utterances. Table 1 shows an example. MRs can be regarded as the multi-conditions type of utterance generation, which consists of slots and values, and the corresponding utterances are references written by humans. We focus on training the model to generate utterances directly from MRs.

MR (slot[value])	name[Giraffe], eatType[pub], food[Fast food], area[riverside], familyFriendly[yes]
Utterance	On the riverside the Giraffe is a Fast food, kid friendly pub.

Table 1: An example of an E2E dataset consists of pairs of a MR and an utterance.

Many of the previous NLG research are a two-stage approach through sentence planning and surface realization. Sentence planning determines the overall sentence structure and surface realization is the process of flattening the sentence structure. In recent studies (Konstas and Lapata, 2013; Dušek and Jurčíček, 2016; Juraska et al., 2018), these two stages are processed at once by learning end-to-end without aligned data with a neural network.

When generating sentences from an input (flat or structured MR), there are a template-based approach and a neural network-based approach. Smiley et al. (2018); Puzikov and Gurevych (2018); Wiseman et al. (2018) generate sentences based on template. The template method is to obtain the structural sets of sentences corresponding to MRs from training data and apply the template appropriate to the test data to generate the sentences. In Smiley et al. (2018); Puzikov and Gurevych (2018), a template is formed based on rules, and Wiseman et al. (2018) learns the template structure of a sentence as a neural network.

There is also a way to generate a natural language with a neural network without using a template. Dušek and Jurčíček (2016); Smiley et al. (2018); Puzikov and Gurevych (2018); Juraska et al. (2018); Elder et al. (2019); Gehrmann et al. (2018) are sequence-to-sequence models of encoders and decoders, which have a one-stage framework, and

Proceedings of Recent Advances in Natural Language Processing, pages 805–813
Sep 1–3, 2021.
https://doi.org/10.26615/978-954-452-072-4_092

Dušek and Jurčíček (2016) is the model used by E2E challenge organizers. In Balakrishnan et al. (2019), the encoder and decoder model converts flat MRs to structure MRs, and outputs the sentence through constrained decoding.

Template and neural network methods each have advantages and disadvantages. The template method guarantees a certain performance but limits the diversity and possibilities of the output. The neural network method generally performs better than the template method, but requires a lot of data and has limitations in the naturalness and semantical correctness of sentences (Nayak et al., 2017).

We propose a novel approach using the Transformer decoder as a simple one-stage framework. In our model, GPT2-small (Radford et al., 2019) is the backbone, and (s_i, v_i) pairs of the meaning representation are put into multiple conditions to generate a sentence. In the previous works (Juraska et al., 2018; Balakrishnan et al., 2019; Smiley et al., 2018; Puzikov and Gurevych, 2018; Dušek and Jurčíček, 2016), when receiving the meaning representation as input, the value is delexicalized. Specifically, all values corresponding to the same slot are delexicalized to a placeholder so that unseen inputs can be processed. However, Nayak et al. (2017); Juraska et al. (2018) report that delexicalization often leads to inappropriate behavior in scenarios. For example, the word ”cheap” can be reflected in the utterance that matches the value of ”less than $20”. Also, when food[Italian] is given as slot[value], ”Italian food” is an appropriate phrase in a generated utterance, but in the case of food[fast food], ”fast food food” is an incorrect phrase. Additionally, the combination of eatType[coffee shop] and food[Italian] is rather weird, and the combination of name[The Rice Boat] and area[riverside] is appropriate, so delexicalization doesn’t fully utilize the characteristics of tokens. We, therefore, treat the slot as a special token and the value as a regular token of vocabulary without delexicalization in the training. Nevertheless, in testing for unseen values, the model generates appropriate utterances and is described in Section 4.3. Our method is considered as generating a sentence as a simple one-stage framework directly from flat MRs.

Our model is tested on the E2E dataset. In addition to the evaluation metrics used in the E2E challenge, the systems are evaluated with BERTScore (Zhang* et al., 2020). Our approach shows the best performance in BLEU, METEOR, and BERTScore and competitive performance in other metrics. By leveraging the pre-trained model GPT2, we quickly converge the model with only a few epochs and generate fluent utterances without considering the structure of the sentence. In addition, even if only 10% of the training data is used, it achieves performance comparable to previous systems.

2 Related Work

In many NLP tasks, the Transformer-based (Vaswani et al., 2017) models have recently shown good performance. Gehrmann et al. (2018) is a previous work using the Transformer encoder and decoder in the E2E task. BERT (Devlin et al., 2019) composed only of the Transformer encoder is used a lot in NLU tasks, and GPT (Radford et al., 2019) composed only of the Transformer decoder is used a lot in NLG tasks. These models are trained as large-scale open-domain corpora. By leveraging pre-trained models trained on large datasets and applying them to downstream tasks, many NLP tasks achieve better performance.

The generation of utterances from MRs is quite similar to machine translation, one of the *sequence-to-sequence* tasks. Also, in terms of generating sentences with certain restrictions, it is similar to style transfer (Logeswaran et al., 2018; Lample et al., 2019; Lee, 2020), which is one of *sequence+condition-to-sequence* tasks. However, the generation of utterances from MRs is not exactly the same as the above tasks because of the *condition-to-sequence* perspective. Since the E2E task does not have a given sequence as the input of the model, we approach the sentence generation using only the decoder without sequence encoding. We choose a language model of the Transformer decoder that performs better than LSTM and uses the pre-trained model GPT2 as a backbone.

3 Our Approach

3.1 Problem Statement

E2E dataset The domain of the E2E dataset is a restaurant and consists of $D = \{(\boldsymbol{M}_1, u_1), \cdots, (\boldsymbol{M}_n, u_n)\}$. $\boldsymbol{M}_i$ is the MR and contains the (slot, value) pair, (s_i, v_i), and u_i is the corresponding utterance. There are 8 types of slots, and the value corresponding to each slot has various numbers (2 to 34). Statistics of the overall dataset are shown in Table 2.

E2E Dataset	MRs	References	Slots/MR	Tokens/Ref
training	4,862	42,061	5.52	20.27
develpoment	547	4,672	6.3	24.52
test	630	4,693	6.91	26.76

Table 2: Statistics of E2E dataset provided by Dušek et al. (2020).

The goal of the task is to generate a u_i by reflecting $\boldsymbol{M}_i$. s_i is the concept of a given category of conditions, and v_i is an item that should actually be reflected in utterance generation. Since utterances can also reflect values as synonyms, changing them to a single placeholder is considered a risk that the processing of input and output will not work properly. In model training, not only delexcalization but also preprocessing of special data such as Smiley et al. (2018) is not performed, and the model is expected to learn about tokens and phrases with different inputs and outputs such as synonyms.

3.2 Pre-trained Model: GPT2-small

Our system uses GPT2-small model as a backbone to generate utterances and is illustrated in Figure 1. GPT2 is an unsupervised pre-trained model with large-scale open-domain corpora of unlabeled text. GPT2 uses only the Transformer decoder and generates sentences from left to right autoregressively without an encoder. GPT2-small has 768-dimensional embedding size, 12 heads, and 12 layers, so the total number of parameters is 117M. Our system has the advantage of starting from knowing the token distribution by using the pre-trained model as the initial state.

3.3 Generation

As conditions of an utterance generation, (s_i, v_i) pairs of a flat MR are given, slots are treated as special tokens. Specifically, the tokens for *slots* and *start* are added to vocabulary, and the regular tokens in vocabulary are used for *values* and *end*. In order to distinguish between special tokens and regular tokens, special tokens are changed to <*SPECIAL TOKEN*> to form vocabulary. Inputs are given by concatenating the given (s_i, v_i) pairs in order (e.g. <*name*>, Giraffe, <*eatType*>, pub, ...). We use <*name*>, <*eatType*>, <*food*>, <*priceRange*>, <*customer rating*>, <*area*>, <*familyFriendly*>, <*near*> in the fixed order of slots as formed in the E2E dataset, and do not put slots not in the given MR as input. Our system generates utterances by considering the MR as multi-conditions of generation. Through training end-to-end Multi-Conditioned generation, we hope that the model find out the role of a MR.

Transformer Decoder is autoregressively unidirectional from left to right, so the model only sees the previous tokens:

$$\boldsymbol{o_i} = \text{TransformerDecoder}(u_i^{1:k}, \text{<}\textit{START}\text{>}, s_i, v_i) \tag{1}$$

where $u_i^{1:k}$ denotes tokens up to k^{th} tokens of u_i. $\boldsymbol{o_i}$ is the output of Transformer Decoder, $k \times h$ tensor, and h is the hidden embedding dimension of the decoder.

To predict the token of the next step $(k+1)$, multiply the k^{th}-vector o_i^k by matrix M as follows:

$$u_i^{k+1} = argmax(M(o_i^k)) \tag{2}$$

where M is a randomly initialized matrix.

Because our system is a one-stage framework that generates utterance directly from flat MRs, sentence planning and surface realization are not considered separately. Our approach shows strong results in 4.2 without additional techniques such as delexicalization, data augmentation, and extra datasets. When testing, it is possible to deal with unseen values by delexicalization, which is described in detail in Section 4.3.

3.4 Training

We experiment using one V100 16GB GPU in Linux environment on an AWS server. Our system is end-to-end trained with the AdamW (Loshchilov and Hutter, 2019) optimizer for 5 epochs. The initial value of the learning rate is 2e-5 and is adjusted with a linear scheduler. The model is trained so that the output at the current step (k) predicts the token of the next step $(k+1)$ and the loss of the objective function is calculated as:

$$\begin{aligned}\mathcal{L}(\theta) = -\sum_{(\boldsymbol{M}_i, v_i)\in D} &(\log p(u^1 | \text{<}S\text{>}, s_i, v_i) \\ &+ \sum_{k=1} \log p(u_i^{k+1} | u_i^{1:k}, \text{<}S\text{>}, s_i, v_i) \\ &+ \log p(\text{<}E\text{>} | u_i, \text{<}S\text{>}, s_i, v_i))\end{aligned} \tag{3}$$

where u_i^k is the k^{th} tokens of u_i and <*S*>, <*E*> are *start*, *end* token respectively. Since the ground truth given in the dataset is only utterance, the outputs before <*S*> is entered cannot be used for loss calculation.

During training time, tokens are generated by applying teacher-forcing, and tokens are generated by self-feeding during testing time.

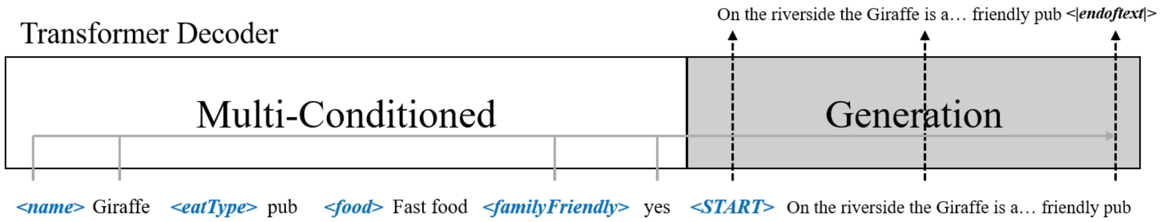
Transformer Decoder
On the riverside the Giraffe is a… friendly pub <|endoftext|>
Multi-Conditioned
Generation
<name> Giraffe
<eatType> pub
<food> Fast food
<familyFriendly> yes
<START> On the riverside the Giraffe is a… friendly pub

Figure 1: Our structure has GPT2 backbone based on the Transformer decoder. Blue tokens are special tokens that are newly added to the vocabulary. The model autoregressively starts to generate utterance from when the <*START*> token is received as input, and ends when the $< |endoftext| >$ token is output.

4 Experiments

We use the model provided by HuggingFace [1] to make it easy to use the pre-trained GPT2 trained by OpenAI.

4.1 Evaluation Metrics

We use the five automatic evaluation metrics used in the E2E Challenge, BLEU (Papineni et al., 2002), NIST (Lin and Och, 2004), METEOR (Denkowski and Lavie, 2014), ROUGE (Lin, 2004) and CIDEr (Vedantam et al., 2015), equally as the basis. Evaluation scripts are provided by challenge organizers [2]. We additionally calculate the similarity F1 scores of the two sentences using the BERTscore of RoBERTa model (Liu et al., 2020) provided by the library [3]. The two sentences entered in the BERTscore library are the generated utterances and human references. The BERTScore metric is task agnostic and, unlike previous metrics, uses importance weighting between contextual embedding. Therefore, it is a common metric that calculates a better correlation by solving the disadvantages of the previous metric. BERTScore is measured only for the systems that provided the output for the test dataset.

4.2 Results

Table 3 shows the experimental results of our system and comparison systems. The first section is our model *Multi-Conditioned Transformer*, which consists of the Transformer decoder.

4.2.1 Compared Systems

TGen (Dušek and Jurčíček, 2016) is the baseline tested by the E2E challenge organizer. SeqGen (Smiley et al., 2018) is the system that participated in the challenge, and Slug2Slug (Juraska et al., 2018) is the system that won the E2E challenge. Slug2Slug improves performance by learning the surface realization model as additional data and ensemble the three models. Model-T (Puzikov and Gurevych, 2018) and TempleGen (Smiley et al., 2018) are rule-based systems using templates. NTemp + AR (autoregressive) (Wiseman et al., 2018) is a hidden semi-markov model (HSMM) decoder that learns the structure of a template. Template-based systems guarantee a certain quality and fluency of natural language generation, but overall performance is lower than neural networks. Dot-copy and Transformer (Gehrmann et al., 2018) are methods of learning the structure of a template with a neural encoder and decoder. The hyperparameter K of these two systems indicates the number of models to be diverse ensembling. TripAdvisor (Elder et al., 2019) follows a two-stage approach: (1) content selection at the system input to generate a symbol intermediate representation and (2) generating utterance. Each stage proceeds with the structure of a neural encoder and decoder and improves the performance of the model with additional data.

4.2.2 Automatic Evaluation

The performance of our system and the comparison systems are shown in Table 3. In these systems, Multi-Conditioned Transformer achieves the best performance in BLEU, METEOR, and BERTScore, and the second-best in NIST. Our system also shows competitive performance compared to previous systems in ROUGE and CIDEr metrics. We experimented with at least 3 random seeds and observe that our model always reaches similar performance.

”No pre-trained” is a model trained from scratch and the rest are the same except for initialization. If our model is trained without the pre-trained tech-

[1] https://huggingface.co/gpt2
[2] https://github.com/tuetschek/e2e-metrics
[3] https://pypi.org/project/bert-score/

System	BLEU	NIST	METEOR	ROUGE_L	CIDEr	BERTscore
Multi-Conditioned Transformer	**0.6794**	8.6477	**0.4579**	0.6998	2.2884	**0.942**
30% sampling (avg)	0.6651 (0.002)	8.5712 (0.035)	0.4364 (0.0056)	0.6871 (0.0022)	2.1561 (0.0360)	0.940 (0.00067)
10% sampling (avg)	0.6541 (0.0024)	8.4332 (0.043)	0.4271 (0.0033)	0.6761 (0.0039)	2.0786 (0.0354)	0.939 (0.00045)
No pre-trained	0.5885	8.0320	0.3962	0.6302	1.7585	0.930
Tgen (baseline)	0.6593	8.6094	0.4483	0.685	2.2338	0.939
Model-T	0.5657	7.4544	0.4529	0.6614	1.8206	0.938
Slug2Slug	0.6619	8.613	0.4454	0.6772	2.2615	**0.942**
TemplGen	0.4202	6.7686	0.3968	0.5481	1.4389	-
SeqGen	0.6336	8.1848	0.4322	0.6828	2.1425	-
NTemp+AR	0.598	7.56	0.3875	0.6501	1.95	-
dot, copy, K = 2	0.674	8.61	0.452	0.708	2.31	-
Transformer, K = 2	0.662	8.6	0.457	0.704	**2.34**	-
TripAdvisor	0.6738	**8.7277**	0.4572	**0.7152**	2.2995	-

Table 3: Automatic metric scores of our and compared systems in the E2E test dataset. Systems are evaluated with BERTscore along with the five metrics used in the E2E challenge. The number in parentheses is the standard deviation. The bold number is a notation for the best performing system.

MR	name[Blue Spice], eatType[pub], area[riverside]
Multi-Conditioned Transformer	**Blue Spice is a pub in the riverside area.**
Tgen (baseline)	Blue Spice is a pub by the riverside.
Model-T	Blue Spice is a pub located in the riverside area.
Slug2Slug	Blue Spice is a pub in the riverside area.
Reference sample	There is a pub Blue Spice in the riverside area.
MR	name[The Mill], eatType[restaurant], food[English], priceRange[less than £20], area[city centre], familyFriendly[yes], near[Raja Indian Cuisine]
Multi-Conditioned Transformer	**The Mill is a family-friendly restaurant that serves English food for less than £20. It is located in the city centre near Raja Indian Cuisine.**
Tgen (baseline)	The Mill is a family-friendly english restaurant in the city centre near Raja Indian Cuisine with a price range less than £20.
Model-T	The Mill is a family-friendly restaurant which serves English food in the price range of less than £20. It is located in the city centre area, near Raja Indian Cuisine.
Slug2Slug	The Mill is a family friendly English restaurant in the city centre near Raja Indian Cuisine. It has a price range of less than £20.
Reference sample	The Mill, Is a restaurant and is family-friendly, cheap and reasonable priced is very good for the family , We provide full English food. Located near Raja Indian Cuisine In the city centre.

Table 4: Comparison of utterances for given MRs. Samples given according to when 3 and 7 (s_i, v_i) pairs are given. In the E2E dataset, the human reference provides several versions but extracts one sample.

model	quality	naturalness
ours	**4.525**	**4.625**
TGen	4.317	4.498
Slug2Slug	4.340	4.545

Table 5: Average of three workers' ratings

nique, we observe that the model performance is quite degraded. However, our approach simply and effectively derives the generalization performance of the model by using only pre-trained LM without using other techniques such as additional data and ensemble techniques.

4.2.3 Human Evaluation

Human evaluation is performed for quality and naturalness as in Dušek et al. (2020); Juraska et al. (2018), and the results are shown in Table 5. Quality is a score for grammatical correctness and whether generated utterance properly reflects given MRs. Naturalness is a rating of the possibility that utterance is written by a native speaker, regardless of the MRs. We randomly sampled 200 samples from our test set and hired 3 workers from Amazon Mechanical Turk [4] to rate them on a scale of 1(bad)-5(good). Slug2Slug is a system that ranked first and second in quality and naturalness, respectively, in the E2E challenge. In human evaluation, our model shows better than baseline TGen and Slug2Slug.

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

4.2.4 Utterance Generation

Table 4 shows the output of the systems. As with BERTscore calculations, other comparative systems with no output provided cannot verify the utterances. When there are three (s_i, v_i) pairs, there is no difference between our system and previous systems. However, as the number of pairs increases, the possible sentence structures vary, so different systems output different utterances. Test data with many pairs is considered to make a difference in automatic evaluation performance. Since our system is based on the Transformer, it can be more robust and general to long-term sequences than LSTM-based systems.

4.2.5 Training with Less Data

The 2nd and 3rd rows of Table 3 are the results of fine-tuning our model by sampling only a small amount of training dataset. For small training data, 10% and 30% of the entire training dataset are randomly sampled. Our model is trained and averaged by performing random sampling three times in consideration of the possibility that the performance of the model may vary according to the statistics of the randomly sampled dataset. We found that the performance of our model was similar even with random sampling. Our system shows a similar level of performance with a small standard deviation according to the sampled data.

As the sampling of the training data of the system increases, the performance of the system improves. However, if we use more than 50% of the training data through many experiments, our system has little improvement in performance. Our approach can leverage the pre-trained language model to take advantage of the background knowledge of sentence generation. Therefore, it is difficult to expect a linear relationship between increasing the number of training data and increasing performance. However, with the effect of background knowledge, our system, which was trained by sampling 10% of the training data, shows performance comparable to previous systems. The performance of the model trained by sampling 30% of the data is similar to that of the Slug2Slug, the system that won the E2E challenge. Building our system with only 30% of the training data and showing good results demonstrates the effectiveness of using the pre-trained model. If a better pre-trained model is used as the backbone, we hope to build an effective model with less data.

4.3 Generation from Unseen Values

Our model is not trained on unseen values, so it can have weaknesses in real applications. Therefore, we introduce a zero-shot generation method through Sim-Delexicalization. Table 6 shows an example of this experiment. The value of <*familyFriendly*> is not treated as unseen value because it only has *(yes or no)*. Our system generates proper utterance through the following two steps for zero-shot generation.

(1) **Sim-Delexicalization**: The given unseen values are replaced with similar values among the lists of value corresponding to the same slot. In the first example of Table 6, ”Blue Man” is replaced by ”Green Man” and ”2.1 out of 5” is replaced by ”3 out of 5”. In the second example we observe that expensive is replaced by high. Also, taking into account the grammatical aspect, if an unseen value containing ”the” in the <*name*> and <*near*> slots are given, the value list containing ”the” is limited as a candidate (and vice versa). There can be several ways to find similar tokens, but we use BERTscore to select the value with the highest score.

(2) **Relexicalization**: Replaced values are changed back to unseen values in generated utterances. ”Green Man” is deciphered as ”Blue Man” and other values proceed as well.

The generated utterances are of appropriate quality from a human perspective. In the previous study, unlike delexicalization of unseen values to one placeholder, we have the difference of converting to similar values. Changing to one placeholder in the test also has the same risk as in training above, so we used the existing list of values to change it to an appropriate value each time. In other words, our system can generate utterances that are suitably customized for a given (s_i, v_i). Rather than using only BERTscore, it may be helpful to find similar words using word embedding techniques such as Glove (Pennington et al., 2014) and FastText (Bojanowski et al., 2017) but this will be left for further study.

4.4 Experiments on a Different Dataset

Our paper focuses on the E2E dataset, but for the possibility of scaling, we do a simple experiment in the WebNLG challenge task (Colin et al., 2016) similar to the E2E dataset with the same approach. The WebNLG dataset is collected from DBpedia, and the train, development, and test datasets are

Slots	<name>, <food>, <customer rating>, <area>, <near>
Unseen values	Blue Man, hot food, 2.1 out of 5, countryside, the school
Delexicalization	Green Man, Fast food, 3 out of 5, city centre, The Bakers
Generated utterance	Green Man is a fast food restaurant in the city centre near The Bakers. It has a customer rating of 3 out of 5.
Relexicalization	Blue Man is a fast food restaurant in the countryside near the school. It has a customer rating of 2.1 out of 5.

Table 6: Zero-shot generation from unseen values. Given unseen values, the system generates an utterance subject to delexicalization of unseen values to similar seen values.

category	**subject**	**property**	**object**
Airport	Aarhus	leaderName	Jacob_Bundsgaard
	Aarhus_Airport	cityServed	Aarhus
Reference	Aarhus airport serves the city of Aarhus who's leader is Jacob Bundsgaard.		

Table 7: Example of the WebNLG dataset. In one sample, the category is fixed as Airport, and multiple values corresponding to (subject, property, object) can be given.

system	BLEU	ROUGE_L	CIDEr	BERTscore
ours	0.2881	0.4859	2.6784	**0.920**
Baseline	0.214	0.3585	1.6754	0.854
Melbourne_nmt	**0.2972**	**0.516**	**3.0245**	0.884
UPF_pipeline	0.2641	0.5018	2.7679	0.883

Table 8: Comparison of systems evaluated with the WebNLG dataset. It was evaluated using the same library as the E2E dataset.

6940, 872, and 1862, respectively, and examples are shown in Table 7. It is significantly smaller than the E2E dataset, and MRs consisting of values corresponding to four (category, subject, property, object) slots are given as a condition. In other words, unlike E2E, the WebNLG dataset has 4 fixed slots, but multiple values can be given.

Table 8 shows the experimental results, and the three comparison systems that participated in the challenge (Gardent et al., 2017) are as follows: (1) The baseline is a neural system trained with OpenNMT [5]. (2) Melbourne shows the best score for all automatic evaluations in the challenge with an end-to-end LSTM with attention model. Performance is improved by preprocessing entity tagging by collecting information from DBPedia. (3) UPF-FORGe (Mille et al., 2017) is a grammer-based NLG system and has the highest score in human evaluation through rule-based graph-transducers for syntacticization.

The systems are evaluated in the same way as the E2E dataset and our system is better than the baseline but slightly worse than the best systems in many metrics. However, our system shows meaningful results in BERTscore and is a simple method that utilizes a pre-trained model without any rule definition, preprocessing, or other datasets. If we preprocess the data like the comparison systems above, our model can expect better performance.

4.5 Analysis

Section 4.3 experimentally demonstrates that our approach is capable of zero-shot generation, a situation not found in the training. The system is trained to generate appropriate utterances for (s_i, v_i) without the delexicalization. Therefore, there is no need to take the ambiguous risk of replacing unseen values with a single delexicalization of placeholders. Instead we introduce sim-delexicalization, which allows the system to reflect unseen values.

The two-stage framework needs to reveal the structure of the sentence, so it is difficult to solve as the number of (s_i, v_i) pairs increases. However, our approach is easily extensible for more pairs. In the WebNLG dataset, we show that it is possible to extend multiple values as well. Since only slots are replaced with special tokens and values are used as regular tokens, our system can be trained to learn the utterance corresponding to MRs without limiting the number of pairs.

We also conducted experiments using a larger backbone model, GPT2-large, but the change in performance is small.

5 Conclusion

This paper presents a simple one-stage approach to generating natural utterances from flat MRs. Our system uses a pre-trained model language model to improve the performance of the system and shows that it is better than the system that won the challenge in human evaluation. Even if our model is trained with only a small amount of sampling data due to the leveraging effect, it is comparable to the previous models. Our approach is simple, efficient, easy to extend to multiple MRs (i.e. WebNLG), and enables zero-shot generation without additional data through sim-delexicalization.

[5] https://opennmt.net/

References

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained decoding for neural NLG from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, pages 831–844. https://doi.org/10.18653/v1/P19-1080.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146. https://doi.org/10.1162/tacl$_{a0}$0051.

Emilie Colin, Claire Gardent, Yassine M'rabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The WebNLG challenge: Generating text from DBPedia data. In *Proceedings of the 9th International Natural Language Generation conference*. Association for Computational Linguistics, Edinburgh, UK, pages 163–167. https://doi.org/10.18653/v1/W16-6626.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*. pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 4171–4186. https://doi.org/10.18653/v1/N19-1423.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 45–51. https://doi.org/10.18653/v1/P16-2008.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language* 59:123–156.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *arXiv preprint arXiv:1901.11528* https://arxiv.org/abs/1901.11528.

Henry Elder, Jennifer Foster, James Barry, and Alexander O'Connor. 2019. Designing a symbolic intermediate representation for neural surface realization. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 65–73. https://doi.org/10.18653/v1/W19-2308.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*. Association for Computational Linguistics, Santiago de Compostela, Spain, pages 124–133. https://doi.org/10.18653/v1/W17-3518.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*. Association for Computational Linguistics, Tilburg University, The Netherlands, pages 46–56. https://doi.org/10.18653/v1/W18-6505.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 152–162. https://doi.org/10.18653/v1/N18-1014.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research* 48:305–346.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1g2NhC5KQ.

Joosung Lee. 2020. Stable style transformer: Delete and generate approach with encoder-decoder for text style transfer. In *Proceedings of the 13th International Conference on Natural Language Generation*. Association for Computational Linguistics, Dublin, Ireland, pages 195–204. https://www.aclweb.org/anthology/2020.inlg-1.25.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81. https://www.aclweb.org/anthology/W04-1013.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*.

Association for Computational Linguistics, page 605.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach. https://openreview.net/forum?id=SyxS0T4tvS.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*. pages 5103–5113.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Bkg6RiCqY7.

Simon Mille, Roberto Carlini, Alicia Burga, and Leo Wanner. 2017. FORGe at SemEval-2017 task 9: Deep sentence generation based on a sequence of graph transducers. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 920–923. https://doi.org/10.18653/v1/S17-2158.

Neha Nayak, Dilek Hakkani-Tür, Marilyn A Walker, and Larry P Heck. 2017. To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *INTERSPEECH*. pages 3339–3343.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. https://doi.org/10.3115/v1/D14-1162.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2E NLG challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*. Association for Computational Linguistics, Tilburg University, The Netherlands, pages 463–471. https://doi.org/10.18653/v1/W18-6557.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1(8):9.

Charese Smiley, Elnaz Davoodi, Dezhao Song, and Frank Schilder. 2018. The E2E NLG challenge: A tale of two systems. In *Proceedings of the 11th International Conference on Natural Language Generation*. Association for Computational Linguistics, Tilburg University, The Netherlands, pages 472–477. https://doi.org/10.18653/v1/W18-6558.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. pages 5998–6008.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 4566–4575.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 3174–3187. https://doi.org/10.18653/v1/D18-1356.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkeHuCVFDr.