Few-Shot Natural Language Generation by Rewriting Templates

Mihir Kale
Google
Mountain View
mihirkale@google.com

Abhinav Rastogi
Google Research
Mountain View
abhirast@google.com

Abstract

Virtual assistants such as Google Assistant, Alexa and Siri enable users to interact with a large number of services and APIs on the web using natural language. The response generation module converts the actions generated by a policy module into a natural language utterance. Traditionally, template based approaches have been used for response generation in virtual assistants. However, such approaches are not feasible for commercial assistants, which need to support a large number of services. Defining templates for a large number of slot combinations for each of the services supported by large scale assistants becomes tedious. In this work, we propose a template rewriting method for Natural Language Generation (NLG), where the number of templates scales only linearly with the number of slots. A set of simple templates is used to convert actions into utterances, which are concatenated to give a semantically correct, but possibly incoherent and ungrammatical utterance. A pre-trained language model is subsequently employed to rewrite it into coherent, natural sounding text. Through automatic metrics and human evaluation, we show that our method improves over strong baselines, while being much more sample efficient.

1 Introduction

Virtual assistants have become popular in recent years and task-completion is one of their most important aspects. These assistants help users in accomplishing tasks such as finding restaurants, buying sports tickets, finding weather etc., by providing a natural language interface to many services or APIs available on the web. Figure 1 shows a general architecture of a task-oriented dialogue system. Most systems include a natural language understanding and dialogue state tracking module for semantic parsing of the dialogue history. This is

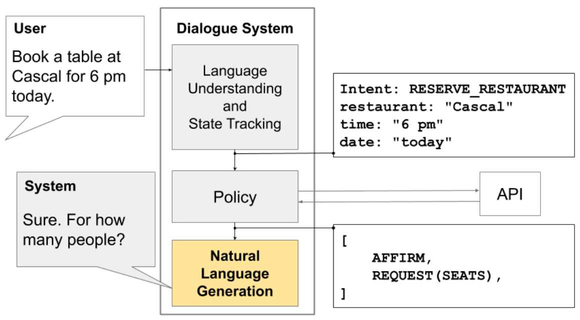
User
Book a table at Cascal for 6 pm today.
Dialogue System
Language Understanding and State Tracking
Intent: RESERVE_RESTAURANT
restaurant: "Cascal"
time: "6 pm"
date: "today"
System
Sure. For how many people?
Policy
API
Natural Language Generation
[AFFIRM, REQUEST(SEATS),]

Figure 1: Typical architecture of a task oriented dialogue system. This work focuses on the Natural Language Generation module.

followed by a policy module which interacts with the APIs, whenever required, and generates the actions to be taken by the system to continue the dialogue. In the end, the Natural Language Generation (NLG) module converts these actions into an utterance, which is surfaced to the user. Being the user-facing interface of the dialogue system, NLG is one of the most important components impacting user experience.

Traditional NLG systems heavily utilize a set of templates to produce system utterances. Although, the use of templates gives good control over the outputs generated by the system, defining templates becomes increasingly tedious as more APIs are added. Supporting multi-domain conversations spanning across multiple APIs quickly grows out of hand, requiring expert linguists and rigorous testing to ensure the grammatical correctness and appropriateness of generated utterances.

Consequently, data-driven generative approaches have gained prominence. Such systems require much less effort and can generate utterances containing novel patterns. Meanwhile, with the rapid proliferation of personal assistants, supporting large number of APIs across multiple domains has become increasingly important,

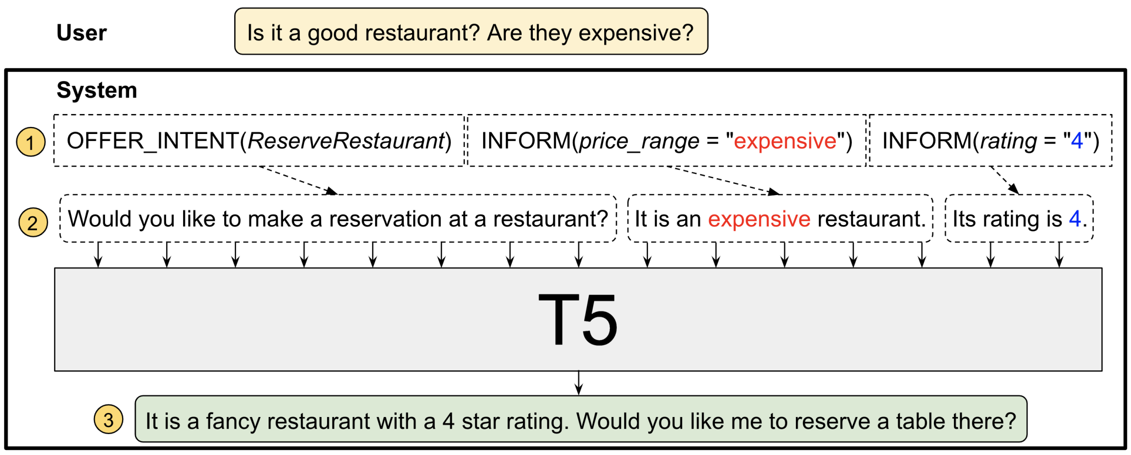
User
Is it a good restaurant? Are they expensive?
System
1
OFFER_INTENT(ReserveRestaurant)
INFORM(price_range = "expensive")
INFORM(rating = "4")
2
Would you like to make a reservation at a restaurant?
It is an expensive restaurant.
Its rating is 4.
T5
3
It is a fancy restaurant with a 4 star rating. Would you like me to reserve a table there?

Figure 2: Overall architecture of our proposed approach. 1. The policy module outputs a set of actions in response to the user utterance. 2. Simple templates convert each action into a natural language utterance. 3. Template-generated utterances are concatenated and fed to a T5 encoder-decoder model(Raffel et al., 2019). The model rewrites it to a conversational response surfaced to the user.

resulting in research on supporting new APIs with few labelled examples (few-shot learning). To this end, generative models pre-trained on large amounts of unannotated text corpus have been increasingly successful.

In this work, we study the use of pre-trained generative models for NLG. Our key contributions are threefold:

1. We propose a simple template-based representation of system actions, and formulate NLG as a utterance rewriting task. We demonstrate the superiority of this approach by automatic and human evaluations.

2. We introduce the SGD-NLG dataset as a benchmark for few-shot and zero-shot learning of natural language generation. Our dataset is based on the SGD dataset (Rastogi et al., 2019) and exceeds all other datasets in terms of number of domains, providing a total of 20 domains across training and evaluation sets.

3. We conduct an extensive set of experiments to investigate the role of dialog history context, cross-domain transfer learning and few-shot learning. We share our findings to guide the design choices in future research.

Our approach achieves state-of-the-art on the MultiWOZ dataset. Next, through experiments on the multi-domain SGD-NLG dataset, we show that this approach enjoys several desirable properties such as strong generalization to unseen domains and improved sample efficiency. Finally, human evaluations show that raters prefer our model's generated responses over human authored text.

2 Related Work

Natural language generation from structured input (NLG) has been an active area of research, facilitated by creation of datasets like WikiBio (Lebret et al., 2016), E2E challenge (Novikova et al., 2017), WebNLG (Gardent et al., 2017) and MultiWOZ (Budzianowski et al., 2018). Neural sequence models have been extensively used in a variety of configurations for NLG in task-oriented dialogue systems. Wen et al. (2017) proposed a two-step approach: first generating a delexicalized utterance with placeholders for slots and then post-processing it to replace placeholders with values from API results, whereas Nayak et al. (2017) highlighted the importance of conditioning generated responses on slot values.

Sequence to sequence architectures directly converting a sequential representation of system actions to system response are also very common (Wen et al., 2015; Dušek and Jurcicek, 2016b; Zhu et al., 2019; Chen et al., 2019a). Domain-adaptation and transfer learning in low resource settings has also been an extensively studied problem (Tran and Le Nguyen, 2018; Chen et al., 2019b; Peng et al., 2020; Mi et al., 2019), with recently released datasets like SGD (Rastogi et al., 2019) and FewShotWOZ (Peng et al., 2020) providing a good benchmark.

Recently, language models pre-trained on large

Approach	Representation of System Actions
Naive	inform (restaurant = Opa!) inform (cuisine = greek)
Slot Description	inform (name of restaurant = Opa!) inform (type of food served = greek)
Template	How about the restaurant Opa!. The restaurant serves greek food.
Ground Truth	Opa! is a nice greek restaurant. How does it sound?

Figure 3: An example showing the representation of system actions utilized by the three schemes. The template representation is generated by concatenating sentences obtained from two templates, which are "*inform(restaurant = \$x)* $\rightarrow$ *How about the restaurant \$x.*" and "*inform(cuisine = \$x)* $\rightarrow$ *The restaurant serves \$x food.*".

amount of unannotated text corpus have achieved state of the art performance across several natural language processing tasks (Devlin et al., 2019; Yang et al., 2019; Liu et al., 2019; Radford et al., 2019; Keskar et al., 2019). Pre-trained generative models have shown promising results for NLG in dialogue systems in low resource settings (Budzianowski and Vulic, 2019; Peng et al., 2020; Kale and Roy, 2020).

Our template based approach bears similarities to the sentence fusion task (Barzilay and McKeown, 2005), where the aim is to combine multiple sentence into a single coherent sentence. While it has been applied to the multi-document summarization task, in this work we demonstrate its effectiveness for task oriented response generation.

3 Model

For a given dialogue turn, let $\mathcal{A} = \{d_i(s_i = v_i)\}_{i=1}^{A}$ be the set of actions which are output by the system, where A is the total number of actions output by the system for this turn. Each action consists of a single dialogue act d_i representing the semantics of the action, along with optional slot and value parameters - s_i and v_i respectively. For example, *inform*, *req_more* and *request* are some of the dialogue acts defined in the SGD-NLG dataset (Rastogi et al., 2019), which are used for informing the value of a slot to the user, asking if the user needs some other help, and requesting the value of a slot from the user respectively. Some acts like *inform* require both the slot and value parameters, whereas acts like *request* require the slot parameter only and acts like *req_more* require none. Some datasets allow multiple slot value arguments for a single act, but such actions can be converted to the above representation by decomposing them into multiple actions with the same act, each containing exactly one slot value pair.

The goal of NLG is to translate $\mathcal{A}$ to a natural language response with the same semantic content. To this end, we first convert the set $\mathcal{A}$ into a sequence (Section 6). Then, we utilize the Text-to-Text Transfer Transformer (T5) (Raffel et al., 2019) model, which is a sequence to sequence model, to generate the natural language response.

Statistics	MultiWOZ	SGD-NLG
#Domains	7	20
#Unseen domains	0	4
#System acts	7	10
#Slots	23	365
#unique values	14K	21K
#train examples	57K	41K

Table 1: Comparison between MultiWOZ and SGD-NLG datasets.

Implementation Details We use the T5-small model which has 6 layers each in the encoder and decoder, with a total of around 60 million parameters. In each of the experiments reported in this paper, we started with a pretrained T5-small model released on its website [1]. The model was then fine-tuned on the corresponding dataset using a constant learning rate of 0.001 and batch size of 256. In all experiments, we observed that the model converged before 1000 steps. The checkpoint yielding the highest BLEU score on the development set was picked for reporting test set results. During inference, we use beam search with a width of 4 and length penalty $\alpha = 0.6$.

4 Datasets

We conduct experiments on 2 datasets - SGD-NLG (Rastogi et al., 2019), MultiWOZ (Budzianowski et al., 2018). SGD-NLG features a larger number of domains and slots as compared to Multi-WOZ and the presence of multiple services per domain makes it representative of practical scale-related challenges faced by today's virtual assis-

[1]github.com/google-research/text-to-text-transfer-transformer

tants. Furthermore, the evaluation sets contain many domains, and consequently slots, which are not present in the training set, to help evaluate model performance on unseen domains. Prior work (Mi et al., 2019; Tran and Le Nguyen, 2018; Wen et al., 2016) has studied zero shot learning, domain adaptation etc. in a simulated setting mainly by holding out domains for adaptation one at a time and creating small subsets. Datasets so far are also very limited in the number of domains. The largest dataset so far, MultiWOZ, has just 5 domains in the test set. Moreover, lack of knowledge of the exact data splits makes it difficult to make comparisons to other methods.
On the other hand, the large size and variety of the SGD dataset makes it a great testbed to study zero-shot learning, few-shot adaptation etc. Having a canonical split will make it easier for future work to compare results across methods.
To encourage reproducible research and a single benchmark that can support different paradigms like joint modeling, domain adaptation etc, we make a new version of the SGD dataset as follows:

- To study few-shot learning from scratch, we make k-shot subsets for varying values of k $[5, 10, 20, 40, 80]$. In this setting each domain has k dialogues.
- For all the few shot splits we make sure that they contain examples for every dialogue act and slot type.
- Multi-domain dialogues are removed from the training data. Though many dialogues are discarded, we found that this led to minimal loss in quality.
- The dev and test sets are left untouched.

We call this dataset SGD-NLG. A comparison with the MultiWOZ dataset can be found in Table 1. The dataset and code will be made publicly available in the future.

5 Evaluation

Automatic Metrics Following prior work, we use BLEU and Slot Error Rate (SER) as automatic metrics. SER represents the fraction of generated texts where at least one slot was not correctly copied from the structured data. Since this metric relies on string matching, we cannot use it to evaluate binary slots like *has_live_music*.

Human Evaluation We conduct a human evaluation study via crowd sourcing. Each worker is shown the dialogue act and responses predicted by the NLG models. Following (Peng et al., 2020), they are asked to rate each response on a scale of 1 (bad) to 3 (good) along two axes - *informativeness* and *naturalness*. Each example is rated by 3 different workers. The final metric is an average of all the ratings.

6 Encoding System Actions

We experiment with three different representations of system actions as shown in Figure 3, and described below.

6.1 Naive Representation

This approach utilizes the most basic representation of actions, similar to that used in Peng et al. (2020). Canonical representations of each action - $a_i, a_i(s_i)$ or $a_i(s_i = v_i)$, depending on the parameters present in the action, are concatenated together to obtain a sequence representation of $\mathcal{A}$. Although this representation is simple to obtain, it suffers from two drawbacks -

(i) **Semantics -** This representation doesn't convey much information about the semantics of a slot. Consequently, the model may need a larger number of training examples to identify the semantics from their usage in the system utterance.

(ii) **Representation Bias -** This representation is very different from what the encoder has seen during pretraining phase, which is natural language text. As a result, the representations learnt during pre-training may not transfer well. Peng et al. (2020) mitigate this by conducting additional pre-training utilizing large scale annotated dialogue datasets. While this method is effective, a large in-domain corpus may not always be available.

6.2 Slot Description Based Representation

Recent work on low-resource natural language understanding tasks have utilized natural language descriptions of slots. These descriptions are easy to obtain, directly encode the semantics of the slot and have been shown to help when in-domain training data is sparse. On similar lines, we extend the Naive representation by replacing the slot names with their natural language descriptions.

Action	Template
notify_success	Your ride is booked and the cab is on its way.
goodbye	Have a safe ride!
request(dest)	Where are you riding to?
request(shared)	Are you comfortable sharing the ride?
confirm(dest=\$x*)*	You are going to \$x.
inform(fare=\$x*)*	Your ride costs \$x dollars.
inform(seats=\$x*)*	The cab is for \$x riders.

Figure 4: Example templates for a ride-sharing API. Parameterized templates are defined for actions which contain a slot value.

The action representations are $a_i, a_i(desc(s_i))$ and $a_i(desc(s_i) = v_i)$, where $desc(s)$ represents a natural language description of slot s. This solves the first drawback of the Naive representation mentioned above. We refer to this method as SlotDesc.

6.3 Template Based Representation

We solve the representation bias problem by converting the set of actions output by the system into a natural language utterance. We employ a technique similar to that used in Rastogi et al. (2019) and define a minimal set of templates. Specifically, as shown in Figure 4, we define one template for each system action. The representation of $\mathcal{A}$ is obtained by concatenating the corresponding templatized representation of each action in $\mathcal{A}$.

Note that, our focus here is not to generate conversational and grammatically correct utterances, but to have a simple representation of the actions, which can be rewritten by the model to a natural and fluent response. Hence, we don't need to cover all edge cases typically required in template based methods - handling of plurals, subject-verb agreement, morphological inflection etc. - and only need to define a small number of templates. For most APIs, this amounts to around 15-30 templates. The actual number varies depending on the number of slots and intents supported by the API. Since this method relies on a combination of templates and transfer learning from language models, we name it **Template Guided Text Generation (T2G2)** .

6.4 MultiWOZ

Baselines Besides our proposed approaches, we also compare with the following baselines:

- **HDSA** - Hierarchically Disentangled Self-Attention (Chen et al., 2019a), a transformer based architecture that exploits the structure of dialog acts to build a multi-layer hierarchical graph.

- **SC-GPT** - A GPT-2 based pre-train + fine-tune approach that relies on a large in-domain NLG corpus. This model currently holds state-of-the-art on MultiWOZ.

- **Copy** - A trivial baseline where the input template is treated as the output text. Though the text is accurate and contains all the information, its likely to sound unnatural and ungrammatical.

	BLEU	SER
Copy	13.12	0.0
HDSA	26.48	12.14
SC-GPT	30.76	**0.53**
Naive	**34.96**	3.60
T2G2	34.91	3.70

Table 2: Performance of models on MultiWOZ.

	Seen		Unseen	
Approach	BLEU	SER	BLEU	SER
Naive	25.87	1.33	15.33	1.18
SlotDesc	25.71	2.0	16.78	0.6
T2G2	**28.34**	**0.75**	**22.03**	**0.09**

Table 3: Performance of models on seen and unseen domains of the SGD-NLG dataset.

The results are shown in Table 2. Naive achieves a BLEU score of 34.96 and outperforms the previous best model SC-GPT by 4+ points, setting a new state of the art. SC-GPT consists of a GPT-2 model further pre-trained on a large in-domain NLG corpus. On the other hand, we do not use any such corpus and directly fine-tune on T5. While the SER score is slightly higher, we found that most of the errors can be attributed to the noisy string matching aspect of the metric and were not actually errors. T2G2 performs on par with Naive. This is likely due to the large size of the MultiWOZ dataset (57K utterances spread over just 5 domains) and indicates that with enough annotated data, a simple pre-train and fine-tune approach is enough to attain good performance. Few shot and zero shot settings offer a greater and more realistic challenge, and we explore these settings next.

Model	**Input or Generated sequence**
Actions	Weather [inform(humidity=28, wind=3)]
Template	The humidity is around 28 percent. The average wind speed should be 3 miles per hour.
Reference	The humidity shows it will only be 28 percent and the wind speed is just 3 miles per hour.
Naive	Its 28 minutes and its 3 minutes.
SlotDesc	The humidity is 28 and the wind speed is 3 miles per hour.
T2G2	The average wind speed should be 3 miles per hour and the humidity is around 28 percent.
Actions	Trains [offer(total=$97, journey_start_time=1:50 pm), inform_count=4]
Template	There are 4 trains that could work. What about the train departing at 1:50 pm? It costs $97 in total.
Reference	There are 4 trains that match your travel plans. There is a train departing at 1:50 pm that will cost $97 in total.
Naive	I found 4 buses for you. The first is from 1:50 pm and the total is $97.
SlotDesc	I found 4 buses for you. The first one departs at 1:50 pm and the total cost is $97.
T2G2	There are 4 possible options. How about a 1:50 pm train that costs $97 in total?
Actions	Media [inform(starring=Andrea Martin)]
Template	Antonio Bustorff acted in it.
Reference	The actor is Antonio Bustorff.
Naive	The movie is directed by Antonio Bustorff.
SlotDesc	The movie is directed by Antonio Bustorff.
T2G2	Antonio Bustorff acted in the movie.

Table 4: A few examples of utterances generated from different models, along with the system actions, their template based representation (as shown in Figure 2) and the reference output. The errors are underlined.

6.5 SGD-NLG

The ideal NLG model should be able to handle domains it was not exposed to during training. In practice, this is very hard to obtain. The SGD-NLG dataset, which features unseen domains in the evaluations sets, let's us asses this zero-shot capability of a model. We report results in Table 3 on two test sets - the *seen* set consists of domains that were seen during training, while the *unseen* set consists of brand new domains aka the zero shot setting. Firstly, all models exhibit low SER scores in both seen and unseen domains, with the template approach being the lowest. This suggests that pre-trained language models are adept at copying and the skill generalizes to out-of-domain examples as well. This result also hints at the need for better evaluation metrics for text generation.

SlotDesc performs on par with Naive on seen domains. At the same time, the slot descriptions do improve performance on the unseen domains (+1.5 BLEU), albeit to a limited degree. More effective ways of incorporating descriptions is a promising are for future work. For the seen domains, template outperforms Naive by 2.7 BLEU. The results on the unseen domain are more striking with template improving on Naive by 6.7 points. This confirms the hypothesis that our simple template based input scheme offers superior generalization capabilities with a low overhead. The template model learns to "fuse" sentences and is able to extend this skill to unseen schemas. The difference in performance between seen and unseen domains, which can be taken as an indicator of the generalization gap, is 12.5 BLEU for the Naive model. T2G2 reduces this to 6.3, effectively halving the gap.

Naturalness				
	Naive	SlotDesc	T2G2	Human
Unseen	2.35	2.45	2.6	2.4
Seen	2.46	2.51	2.51 †	2.38
Overall	2.4	2.48	2.55 *†	2.39
Informativeness				
	Naive	SlotDesc	T2G2	Human
Unseen	2.33	2.36	2.52	2.48
Seen	2.45	2.41	2.47 *†	2.39
Overall	2.39	2.39	2.5 *†	2.44

Table 5: Results of human evaluation. † Significantly better than *human*. * Significanly better than *SlotDesc*. Statistical significance computed using a one tailed t-test, $p < 0.01$

Qualitative Analysis In Table 4 we list a few examples of model predictions. The first one is from Weather, a domain that is not present in the training set. As this domain is not present in the training set, the Naive model completely fails as expected. SlotDesc, on this other hand is able to successfully utilize the descriptions of the domain and slots to produce a largely accurate and fluent response. It, however, misses the word 'percent' when talking about the humidity value. T2G2, which relies on simple, crude templates is able to take that information and rewrite it into a coherent response fully conveying the desired meaning.

The second example is also from an unseen domain - Trains. While Naive and SlotDesc correctly convey the slot related information, they talk about a bus instead of train, since the input most closely resembles the Bus domain seen during training. T2G2, on the other hand, produces accurate text.

The final example illustrates a case where the model has to deal with a seen domain (Media) but an unseen slot (*starring*). This is likely to be a common scenario, where new functionality needs to be added to an existing API. Here, both Naive and SlotDesc incorrectly treat the slot value Antonio Bustroff as a director, since the slot *directed_by* appears in training. T2G2, however, is able to correctly ground the generated text in the input templates to generate the phrase *acted in*.
We refer the reader to the appendix for more qualitative examples.

Human Evaluation We conduct a human evaluation study as described in 5. A total of 500 examples are rated - 250 each from seen and unseen domains - across the 3 models discussed above and the ground truth response (*human*). With 3 ratings per example, this leads to a total of 6,000 ratings. In each rating task, the raters were asked to rate the responses generated by each model and the ground truth response with categorical scores 1 (bad), 2 (average) and 3 (good). For each example, the responses were shuffled in a random order to prevent positional bias among the raters.

From the results in Table 5, we find that on *seen* domains all models perform comparably. Somewhat surprisingly, for seen domains even the baseline model performs on par with the ground truth. This is in line with recent work on task oriented NLG (Peng et al., 2020; Kale and Roy, 2020) which found that large pre-trained models can be fine-tuned to generate responses that match or exceed the data they were trained on. For unseen domains, T2G2 provides large improvements over baselines for both informativeness and naturalness, confirming the trends from automatic improvements. Remarkably, T2G2 also outperforms the human authored ground truth responses. We take this as a promising indication of the real world applicability of our approach.

7 Other Experiments

Even after narrowing down on the choice of a model architecture, there are many possible choices to be made. In this section, we conduct a thorough analysis of these choices and report our empirical findings on different NLG datasets. We hope that these experiments will guide design choices in the future NLG models.

7.1 Few-shot NLG

Virtual assistants need to support a large number of domains and APIs. Supporting new APIs with ease and without the requirement for large amounts of annotated data is very important. In these experiments, we study the impact on performance of NLG models with the amount of available training data. For a k-shot setting, we sample k dialogs from each domain for training. Since there are 14 domains in the train set of SGD-NLG, for a 5-shot setting this will correspond to 70 dialogs. The dev and test sets are untouched. We run experiments by for k in $[5, 10, 20, 40, 80]$. The dataset is referred to as FewShotSGD and we will make the exact splits publicly available in order to facilitate easy and fair comparisons in future research.

Results are reported in Table 6. In all k-shot settings, T2G2 gives consistent improvements of 3-4 BLEU while reducing the SER by 50%. Even in the extreme 5-shot setting, the SER is just 2.66%. Remarkably, T2G2 in the 20-shot setting (280 dialogs total) performs on par with the Naive model trained on the *entire* dataset which is 20X larger (5,403 dialogs). We take this as evidence that our templatized input representation can lead to significant reduction in labelled data requirements.

7.2 Joint Modeling

Domain or API specific models effectively have a higher number of parameters per API, which increases the model capacity. On the other hand, parameter sharing effectively increases the amount of supervision per parameter by utilizing training examples from all APIs. Hence, joint modeling could be beneficial in low resource settings if there is some similarity between the underlying structure. Furthermore, joint modeling also reduces the maintenance workload and is resource efficient. For NLG systems, it could also help in maintaining consistent styles across domains and APIs.

Because of these merits, we investigate the effect of joint modeling for NLG. The SGD-NLG dataset, which features a variety of domains, offers an excellent testbed for such a study. We focus on the 12 domains that are present in all 3 splits - train, dev and test. Concretely, we train a single model on domains and compare it with individual

Model	Metric	5-shot	10-shot	20-shot	40-shot	80-shot	All Data
Naive	BLEU	18.66	20.12	21.26	23.48	24	24.71
	SER	5.91	4.19	3.87	2.01	1.63	1.31
T2G2	BLEU	22.93	24.37	25.75	26.86	27.31	27.68
	SER	2.66	2.4	1.44	1.11	1.04	0.67

Table 6: Performance in few-shot settings. K-shot denotes K dialogues for an API in the training set.

Domain	Separate		Joint	
	BLEU	SER	BLEU	SER
Homes	24.38	0.78	27.92	0.46
Buses	19.06	3.46	24.24	0.24
Media	31.51	3.85	31.19	0.00
RideShare	21.1	1.00	24.78	0.08
Movies	23.39	21.06	33.11	0.15
Flights	19.88	0.67	22.02	0.00
Music	26.57	0.36	28.07	0.00
Services	26.85	0.66	27.08	0.62
RentalCars	19.85	9.08	27.67	0.00
Restaurants	26.82	3.37	28.2	1.18
Events	31.12	0.35	31.26	0.06
Hotels	28.14	3.47	30.08	3.55
Average	24.89	4.01	**27.97**	**0.53**

Table 7: Joint vs domain-specific (separate) NLG.

	Naive		T2G2	
k	BLEU	SER	BLEU	SER
0	24.71	1.31	27.68	0.67
1	25.75	1.31	28.12	**0.45**
3	27.77	**1.28**	30.04	0.51
5	28.43	1.43	30.51	0.60
7	**28.45**	1.47	**30.68**	0.72

Table 8: Changing the size of the context. *k* represents the number of previous utterances used.

models trained for each domain separately. The results are shown in Table 7. We notice consistent improvements in both metrics across all domains. The largest improvement is in the Movies domain, where BLEU improves by 10 points and SER reduces from 21.06 to just 0.15. On an average joint modeling improves BLEU by 3 points and reduces SER from 4% to just 0.53%, demonstrating successful knowledge sharing across domains.

7.3 Role of Context

Dialogue acts represent the semantic content of the system response, but they don't contain any information about the lexical and syntactic content. The utterances in the dialogue context are also important to generate good responses because they can help model conversational phenomena such as entrainment (lexical and syntactic alignment of responses), and can help it avoid repetition (Dušek and Jurcicek, 2016a). Context also helps add variations to the responses generated across different conversations for the same system actions.

Table 8 shows the performance of NLG as more utterances from the dialogue context are given as input. In these experiments, we concatenate the last k utterances to the system action representation obtained from the Naive and template based methods. Both models, Naive and T2G2, benefit from the additional context, showing an improvement of 3-4 BLEU points.

However, we would like to point out that the evaluation is not completely fair, because we used the ground truth system utterances in the context during evaluation as opposed to the utterances generated by the NLG model itself. Regardless, the improvements clearly point to effectiveness of the added context. We hope these results inspire more work in this exciting direction.

8 Conclusions

In this work, we propose a template based input representation for task oriented response generation. Coupled with pre-trained language models, this approach enables zero shot generalization to new domains with little effort. Moreover, we show that it can lead to drastic reduction in annotation costs. We also present the first set of results on the multi-domain SGD-NLG dataset, which we hope will pave the way for further research in few-shot, zero-shot and multi-domain language generation.

While in this paper we use standard pre-trained models, designing pre-training tasks tailored to *sentence fusion* is an interesting line of future work. We also hope to apply T2G2 to languages other than English. Obtaining annotated data in non-English languages is an even bigger challenge, making the sample efficiency of our template rewriting approach especially suited to this setting.

References

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Paweł Budzianowski and Ivan Vulic. 2019. Hello, its gpt-2-how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. *EMNLP-IJCNLP 2019*, page 15.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019a. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709.

Zhiyu Chen, Harini Eavani, Yinyin Liu, and William Yang Wang. 2019b. Few-shot nlg with pre-trained language model. *arXiv preprint arXiv:1904.09521*.

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)*, pages 4171–4186.

Ondřej Dušek and Filip Jurcicek. 2016a. A context-aware natural language generator for dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 185–190.

Ondřej Dušek and Filip Jurcicek. 2016b. 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)*, pages 45–51.

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*, pages 124–133.

Mihir Kale and Scott Roy. 2020. Machine translation pre-training for data-to-text generation–a case study in czech. *arXiv preprint arXiv:2004.02077*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

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

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3151–3157. AAAI Press.

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

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. *arXiv preprint arXiv:2002.12328*.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Van-Khanh Tran and Minh Le Nguyen. 2018. Adversarial domain adaptation for variational neural language generation in dialogue systems. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1205–1217.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2019. Multi-task learning for natural language generation in task-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1261–1266.

Appendix

Sample utterances generated using the different models for various domains are shown in the examples below. The system actions, its template based representation used by the T2G2 model as input, and the reference response are also provided. The predictions are from models trained on the full SGD-NLG dataset and without any dialogue history context. The unseen domains have been marked with an asterisk.

Domain	**Model**	**Input or Generated sequence**
RentalCars	*Actions*	request(start_date=March 12th) request(pickup_location)
	Template	Do you want to have the car from March 12th? Where do you want to pick up the car?
	Reference	Will you be renting the car from March 12th? Please provide the pick-up location as well.
	Naive	When do you want to start on March 12th and where do you want to pick it up?
	SlotDesc	When do you want it from March 12th to pick it up?
	T2G2	Do you want to pick up the car from March 12th and where do you want to pick it up?
Hotels	*Actions*	inform(smoking_allowed=False) notify_success
	Template	Successfully booked the hotel. No, smoking is not allowed on the property.
	Reference	Your reservation is booked. No, smoking is not allowed on the property at this hotel.
	Naive	Your reservation has been made. Unfortunately, the hotel does not allow you to make the reservation.
	SlotDesc	Your reservation has been made. Unfortunately, they do not allow smoking.
	T2G2	I have booked the hotel. No, smoking is not allowed on the property.
Homes	*Actions*	confirm(property_name=Almaden lake Apartments) confirm(visit_date=March 13th)
	Template	Please confirm the following details: You are scheduling a visit to Almaden Lake Apartments and you want to visit the property on March 13th.
	Reference	Please confirm details to schedule a visit to Almaden lake Apartments on March 13th.
	Naive	Please confirm the following details: You are scheduling a visit to Almaden lake Apartments on March 13th.
	SlotDesc	Please confirm: You want to visit Almaden lake Apartments on March 13th.
	T2G2	Please confirm the following details: You are scheduling a visit to Almaden Lake Apartments on March 13th.
Services	*Actions*	request(type)
	Template	What kind of therapist do you need?
	Reference	Let me know your required type of Therapist to search for.
	Naive	What kind of doctor are you looking for?
	SlotDesc	What kind of therapist are you looking for?
	T2G2	What kind of therapist do you need?
Restaurants	*Actions*	confirm(restaurant_name=Nizza La Bella) confirm(location=Albany) confirm(time=6:15 pm) confirm(number_of_seats=2) confirm(date=March 7th)
	Template	Please confirm the following details: Booking a table at Nizza La Bella. The city is Albany. The reservation is at 6:15 pm. The reservation is for 2 people. The date is March 7th.
	Reference	Please confirm: You want me to book a table for 2 people at Nizza La Bella located in Albany for 6:15 pm on March 7th.
	Naive	Please confirm the following details: Booking a table at Nizza La Bella in Albany at 6:15 pm for 2 people on March 7th.
	SlotDesc	Please confirm the following details: Booking a table at Nizza La Bella in Albany at 6:15 pm for 2 people on March 7th.
	T2G2	Please confirm the following details: Booking a table at Nizza La Bella in Albany at 6:15 pm on March 7th for 2 people.
Media	*Actions*	inform(starring=Antonio Bustorff)
	Template	Antonio Bustorff acted in it.
	Reference	The actor is Antonio Bustorff.
	Naive	The movie is directed by Antonio Bustorff.
	SlotDesc	The movie is directed by Antonio Bustorff.
	T2G2	Antonio Bustorff acted in the movie.
Music	*Actions*	confirm(track=strangers) confirm(device=Patio)
	Template	Please confirm the following details: playing the song Strangers in Patio
	Reference	Please confirm: play strangers in patio?
	Naive	Please confirm the following details: Play strangers on the Patio.
	SlotDesc	Please confirm: play strangers on Patio.
	T2G2	Please confirm the following details: playing Strangers in Patio.
Buses	*Actions*	request(from_city) request(departure_date)
	Template	Where are you leaving from? When do you plan to leave?
	Reference	What is your departure date and city?
	Naive	Where are you leaving from and when do you plan to leave?
	SlotDesc	Where are you leaving from and when do you plan to leave?
	T2G2	Where are you leaving from and when do you plan to leave?

Domain	Model	Input or Generated sequence
Events	*Actions*	confirm(event_name=Acting For Teens) confirm(number_of_tickets=4) confirm(date=March 11th) confirm(city=New York)
	Template	Please confirm the following details: Booking tickets for Acting For Teens. You have 4 tickets in your reservation. The date of it is March 11th. The event is taking place in New York.
	Reference	Please confirm: You want me to purchase 4 tickets for the event called Acting For Teens taking place in New York on March 11th.
	Naive	Please confirm: 4 tickets to Acting For Teens on March 11th in New York.
	SlotDesc	Please confirm: 4 tickets to Acting For Teens on March 11th in New York.
	T2G2	Please confirm: Booking 4 tickets for Acting For Teens on March 11th in New York.
Events	*Actions*	confirm(event_name=Mxmtoon) confirm(number_of_tickets=2) confirm(date=March 10th) confirm(city=New York)
	Template	Please confirm the following details: Booking tickets for Mxmtoon. You have 2 tickets in your reservation. The date of it is March 10th. The event is taking place in New York.
	Reference	Can you please confirm that you want to buy 2 tickets for the Mxmtoon event which happens on March 10th in New York?
	Naive	Please confirm: 2 tickets for Mxmtoon on March 10th in New York.
	SlotDesc	Please confirm: 2 tickets to Mxmtoon on March 10th in New York.
	T2G2	Please confirm: 2 tickets for Mxmtoon on March 10th in New York.
Alarm[*]	*Actions*	offer(alarm_time=4 pm) offer(alarm_name=Event)
	Template	There is an alarm for 4 pm with name Event.
	Reference	A 4 pm alarm with name Event.
	Naive	There is an event at 4 pm.
	SlotDesc	There is an event at 4 pm.
	T2G2	There is an alarm for 4 pm with event name.
Alarm[*]	*Actions*	offer(alarm_time=10:30 am) offer(alarm_name=commute) inform_count(count=1)
	Template	You have 1 alarms currently set. There is an alarm for 10:30 am with name Commute.
	Reference	You have 1 alarm at 10:30 am. with name commute
	Naive	I found 1 suitable for you. It's at 10:30 am and it's a commute.
	SlotDesc	I found 1 alarm at 10:30 am. It's a commute.
	T2G2	You have 1 alarm for 10:30 am with the name Commute.
Weather[*]	*Actions*	inform(humidity=30)
	Template	The humidity is around 30 percent.
	Reference	The humidity is about 30 percent.
	Naive	It's 30 minutes.
	SlotDesc	The humidity is 30.
	T2G2	The humidity is around 30 percent.
Weather[*]	*Actions*	inform(humidity=75) inform(wind=4)
	Template	The humidity is around 75 percent. The average wind speed should be 4 miles per hour.
	Reference	The humidity for today is around 75 percent. The average wind is 4 miles an hour.
	Naive	It's a 7 star, and it's a 4 star.
	SlotDesc	The humidity is 75 and the wind speed is 4 miles per hour.
	T2G2	The average wind speed should be 4 miles per hour and the humidity is around 75 percent.
Messaging[*]	*Actions*	notify_success
	Template	You've successfully shared the location.
	Reference	The location was successfully shared.
	Naive	Your reservation has been made.
	SlotDesc	Your contact has been confirmed.
	T2G2	Your location has been shared.
Messaging[*]	*Actions*	confirm(location=2190 Bancroft Way) confirm(contact_name=Peter)
	Template	Please confirm the following details: You're sharing the location 2190 Bancroft Way with Peter.
	Reference	So share the location 2190 Bancroft Way with Peter?
	Naive	Please confirm the following details: You are scheduling a visit to Peter at 2190 Bancroft Way.
	SlotDesc	Please confirm the following details: You would like to contact Peter at 2190 Bancroft Way.
	T2G2	Please confirm the following details: You're sharing the location 2190 Bancroft Way with Peter.
Trains[*]	*Actions*	offer(journey_start_time=7:10 am) offer(total=$91)
	Template	What about the train departing at 7:10 am? It costs $91 in total.
	Reference	There is a train leaving at 7:10 am and costs $91.
	Naive	There is a bus that departs at 7:10 am and costs $91.
	SlotDesc	There is a 7:10 am train that costs $91.
	T2G2	How about the 7:10 am train? It costs $91 in total.
Travel[*]	*Actions*	offer(attraction_name=BODY WORLDS London) offer(category=Museum)
	Template	You should check out BODY WORLDS London. This is a Museum.
	Reference	I suggest a museum called BODY WORLDS London.
	Naive	BODY WORLDS London is a Museum.
	SlotDesc	BODY WORLDS London is a museum.
	T2G2	BODY WORLDS London is a museum.