

Applied Deep Learning Final

吳兩原[†], 蔡宜倫*, 李高迪[†], 謝昊儒*
[†] 電機系* 資工系

National Taiwan University
{b06901032, b06902135, b06901118, b06902018}@ntu.edu.tw

1 Abstract

In this project, two tasks are achieved: NLG(natural language generation) and DST(dialogue state tracking). For NLG, our goal is to decide whether to generate engaging chit-chat response before or after task-oriented response in each system turn with its dialogue context; on the other hand, DST works on given dialogue history (i.e., previous turns) as context, the model could accurately predict the belief states (i.e., a list of (domain, slot, value) triplets), the action decisions (i.e., a list of (domain, action_type, slot) triplets) and a corresponding system response that is functionally accurate and socially engaging for each system turn.

We mainly fine-tune pre-trained model T5 for both tasks and found that the size of the pre-trained model and post-processing makes a huge impact on the performance. Also, we discussed the generated utterances of different pre-trained models between GPT-2 and T5 and showed why T5 is chosen for our final decision. To conclude, we achieved rank ??? in NLG and top 5 performance in DST for both seen and unseen domain.

2 Introduction

Recent years, due to the increasing computing power, a growing number of datasets and modeling innovations, the performance of both task-oriented dialogue systems and chit-chat systems have surged. Most research on dialogue systems focuses on a particular type: TOD(task-oriented dialogue systems aims to accurately track user goals in order to better achieve functional goals) and chit-chat systems(which focuses on user experience, making the conversation more engaging).

For both NLG and DST tasks, we train our models on multi-dataset {dstc8-schema-guided-dialogue [1] + MultiWOZ 2.2 [2] + accentor [3]}. Leveraging the ACCENTOR[3] dataset, we hope to train a virtual assistant capable of adding casual and contextually relevant chit-chat between the given task-oriented dialogues. Hypothesizing that using the chit-chat dialogues could make the assistant more engaging, interesting, human-like and knowledgeable, without being misleading and inappropriate compared to the given original task-oriented dialogues.

For NLG task, we use a language model to generate response including task-oriented and chit-chat response. While the input of the model is the chatting history between user and system, the language model would then output the next turn. Since the generated utterance could be misleading or inappropriate, we further train a filter model which is capable of selecting the generated utterance that could make the overall dialogue more engaging, interesting and human-like.

For DST task, we also use a language model to solve the problem. The input of the model is not only the dialogues between the user and the system, but also previous state (previous slot values corresponding to each possible slot names) in order to tackle the situation when there are unseen slot pairs that have not been seen during training but occurring in testing. The output of the model is the new state (updated slot values corresponding to each possible slot).

3 Approach

3.1 NLG

We referred to this paper: Adding chit-chat to Enhance Task-oriented dialogues[3]. We treat the dialogue contexts as the input of an off-the shelf pre-trained model, and the model will determine the utterance of

the next turn. Also, we add 3 special tokens, "`<|user|>`", "`<|system|>`", and "`<|chitchat|>`". We hope the model can generate the corresponding sentences regarding to the token, whether only the system responses or also with the chitchat. We chose the GPT-2 and T5 as the pre-trained model. After the generation, we trained a Filter model to decide whether we should add the generated chit-chat. We chose bert-base-cased as the pre-trained model.

3.2 DST

We referred to this paper: A Simple Language Model for Task-Oriented Dialogue[4]. The method in the paper is to utilize the GPT-2 model to solve the DST problem and generate responses. The input will be the dialogue contexts, and the output would be the slot's name and value, and the response.

However, in our task there exists unseen domain, and the model can't generate slot pairs that it didn't see before. To solve this problem, we modify the input format: apart from dialogue contexts, we also add previous state's information to the input, including every possible slot names and values, thus allowing the model to copy the slot name from the input to output the unseen domain slot.

The pipeline of DST is shown in Fig. 1. We can find all possible slot names to each service from schema.json. We then input the dialogue contexts and the information of the previous state to the model (in the first turn, the value of previous state would be set to None). For training, the model's input is the dialogue contexts plus the ground truth of previous states (similar to the idea of teacher forcing), while the ground truth of model's output would be the value of each possible slot in the next state. For testing, we use the output of previous turn as the input of next turn, and the final answers would be the output of the last turn.

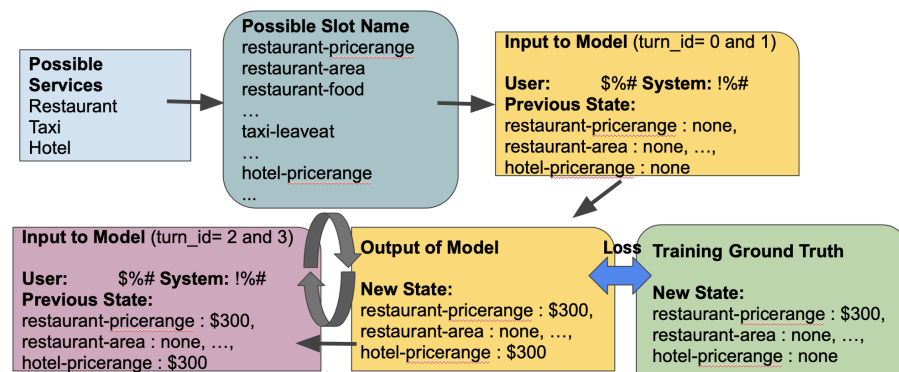


Fig. 1. Pipeline of DST

4 Experiments

4.1 NLG

We first tried GPT-2 model. We first pre-process the training and testing data to the format in Fig. 2 and Fig. 3. We assigned a special token "=", after the special token is the chit-chat we want the model to generate.

However, we found that the result is not good enough. As the example in Fig. 4, the model generate the chit-chat "Do you have any idea what time it is?" This kind of generation should not be the chit-chat, but more like the system's response. We have tuned many times but still can't have better results, so we then tried another model.

We then tried T5 model, and changed a little in the pre-processing, adding another special token "`<|blank|>`", the data format is in Fig. 5. The model will learn to add chit-chat to the position of the "`<|blank|>`" token.

```

artist? And what year was it released?<|system|>The song is by Blues Saraceno and it was released in
2018.<|user|>Thank for the info!<|chitchat|> You are welcome.have a great night .<|system|>Is there
anything else you need?<|endoftext|>",
"chit-chat": [
  " You are welcome.have a great night ."
]

```

Fig. 2. The format of training data

```

"dialogue_id": "SNG01873",
"context": "<|user|>I need to be picked up from pizza hut city centre after 04:30",
"system": "<|system|>Where will you be heading to ?"

```

Fig. 3. The format of testing data

```

<|user|>Do you want me to pick up my car from Pizza Hut City Centre<|chitchat|> Do you
have any idea what time it is?<|system|> I don't know how many cars there are in the ci
ty.<|user|> do you think that sounds like a good place to pick up your car?<|user|> Are
you sure about that?

```

Fig. 4. The result of GPT-2

```

artist? And what year was it released?<|system|>The song is by Blues Saraceno and it was released in
2018.<|user|>Thank for the info!<|blank|>Is there anything else you need?<|blank|>",
"chit-chat": "<|chitchat|> You are welcome.have a great night .<|system|>Is there anything else you
need?</s>"

```

Fig. 5. The format of data in T5

```

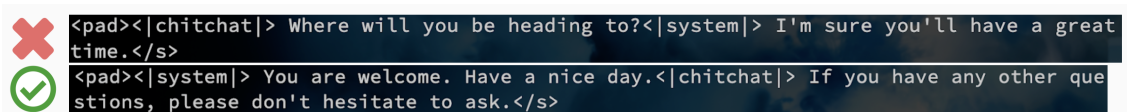
<pad><|system|> You are welcome. Have a nice day.<|chitchat|> If you have any other que
stions, please don't hesitate to ask.</s>

```

Fig. 6. The result of T5

The result of T5 model is way better than GPT-2 model. We found the result will be more like a human response. For example, in Fig. 6., the chit-chat "If you have any other questions, please don't hesitate to ask." is very reasonable.

Finally, we compare the result of adding the Filter model, and found that with the help of the Filter model, we can remove some chit-chats that are not reasonable. For example, in Fig. 7., the generated chit-chat "Where will you be heading to?", which is not related to the afterward system response, is removed by our Filter model.



```

✗ <pad><|chitchat|> Where will you be heading to?<|system|> I'm sure you'll have a great
time.</s>
✓ <pad><|system|> You are welcome. Have a nice day.<|chitchat|> If you have any other que
stions, please don't hesitate to ask.</s>

```

Fig. 7. The result of applying the Filter model

4.2 DST

Experiment details We leverage huggingface's example script run_summarization.py for both training and testing. For all experiments, we only train for 1 epoch due to time constraint. Learning rate is set to 5e-5 with linear decay schedule. Batch size is set to 4. Due to memory constraint, the maximum length for input and output is set to 384 and 256 respectively. It is noted that >99% of training example do not exceed the maximum length. In testing, we use greedy decoding to generate the output.

Model choice The model we first tried is multilingual T5-small. It already achieved 27% joint accuracy on dev set. Since our data is in English, we switched to T5-small, which is only trained with English data. Performance on dev set increased from 27% to 48%. The large performance gap between MT5 and T5 with the same model size is probably due to the high complexity of the task, which requires a thorough understanding of language. We also utilized larger T5 models: T5-base and T5-large, performance on dev set increased to 57% and 62% respectively. It is believed that using larger model such as T5-3b and T5-11b will further improve the performance. The performance of each model can be found in Table 1.

Table 1. DST result with different models

	Test Seen	Test Unseen	Dev set
MT5-small	0.13972(*)	0.09363(*)	0.27
T5-small	-	-	0.48
T5-base	0.30865	0.22471	0.57
T5-large	0.34619	0.26591	0.62

*: Does not involve any post-processing, and only consider previous 3 turns of utterances, while other results involve post-processing and consider previous 5 turns of utterances.

Dialogue context We suspect that the information needed to update slot value may not be all included in 2 turn of dialogue (i.e. 1 SYSTEM-USER pair). Therefore, we experimented adding more dialogue contexts to model input. We found that when only using 2 turn of dialogue, the performance on the dev set is only 41%, but if we add to use 5 turn of dialogue (i.e. SYSTEM-USER-SYSTEM-USER-SYSTEM), the performance would increase to 52%. However, adding more turns of dialogues will increase both memory usage and training / inference time. Also, the model may be more prone to overfitting. The comparison of model performance on using different number of previous turns is shown in Table. 2.

Table 2. DST result with different number of previous turns used

	Test Seen	Test Unseen	Development set
Previous 2 turns	-	-	~0.41
Previous 3 turns	0.26694	0.19101	~0.48
Previous 5 turns	0.27424	0.20224	~0.52

Postprocessing After training the model, we also do some post-processing. There are 2 types of slots: categorical and non-categorical slots. For non-categorical slots, slot value can be extracted from the dialogue; however, for categorical slots, possible slot values are pre-defined and may not appear in the context. Since most slots are non-categorical, our model mostly learn to output values that appear in the dialogue, resulting in low accuracy for categorical slots. To partially solve this problem, we calculate the similarity between the output slot values and all the pre-defined slot values, and then choose the pre-defined slot value that is most similar to the output slot value as the final answer. The metrics we used for comparison are longest common substring and pretrained GLOVE embedding. The performance gains of using different metrics are listed in Table 3.

Table 3. DST result with different receptive field.

	W/O post-processing	Longest Common Substring	GLOVE Embedding
Test Seen	0.32533	0.34619	0.35245
Test Unseen	0.24719	0.26591	0.26591

5 Discussion

5.1 DST

For future work, we may further improve our DST model by

1) Handle categorical and non-categorical slots in different ways

We suspect that the discrepancy between categorical and non-categorical slots is a stumbling block to model training since the model may struggle whether to copy context from the dialogue. For categorical slots, instead of only putting previous slot values into model’s input, we can also feed all possible answers of that slot into the model, and let the model select the correct answer from the several choices. In this way, it is possible to directly answer multiple-choice questions correctly instead of relying on post-processing. Also, the model will not be confused by the difference between the two slot types.

2) Separate service selection and slot tagging into two stages

That is, in the 1st stage, the model receives utterances and possible service names as input, and then output the most probable service involved. Then in the 2nd stage, the model receives utterances and slots related to selected service as input, then output slot values. In this way, length of input and output sequence can be significantly reduced.

In this work, the only information we leverage from the dataset is the services and states of dialogues. We look forward to further explore the potential of a single generative model (without any classification or extractive QA objective) by using additional information in the dataset.

5.2 GPU

During our experiment, the performance of RTX 3070 was below RTX 2080S and in some circumstances even below RTX 2070S, we looked into the specifications of the hardware information, compared to RTX 20 series, the RTX 30 series feature more than twice the amount of the CUDA cores than those in the RTX 20 series, but with slower clock (both base and boost clock). During training and predicting, the GPU utilisation on the RTX 3070 machine was only at approximately 60 percent, with power draw of only 150 watts, indicating that it is not running at full potential (full power should be at 220 watts), the main reason of this is due to not having all of the CUDA cores used, using hardware monitor, we have found out that out of all the CUDA cores, only about 3000 were active, after some optimisation to the model, the model did not seem like using all of the cores, using only about 3500 CUDA cores approximately. Because of that, a 2080S which features 3072 CUDA cores can run faster than a 3070. This could be that later generation are more optimised for games, take "1080ti vs 2080S" for example, 1080ti runs faster than 2080S in training (about 2 percent faster), while 2080S runs faster than 1080ti in games (about 15 percent faster), this could be the cause. However, we cannot eliminate the possibility of hardware issues (faulty graphics card, power connection, bus etc.)

6 Conclusion

This project implemented methods to complete the designated two tasks, which are fine-tuning language models to complete NLG and DST. For NLG, we compared the generated chit-chat from GPT-2 and T5, due to the generated text we decided to use encoder-decoder based T5, also utilizing the filter model to help us better select the generated utterance and achieved rank 1 on engaging and knowledgeable, rank 2 on humanlike and rank 4 on interesting after human evaluation. For DST, we fine-tuned language model T5 to generate new value of each slot which is the output after feeding dialogue to the model and achieved rank 3 in seen domain and rank 4 in unseen domain on Kaggle.

7 Work Distribution

Our work distribution is in Table 4. Two of us focus on NLG and two of us focus on DST.

Table 4. Work Distribution

Work Distribution			
吳兩原	蔡宜倫	李高迪	謝昊儒
NLG	NLG	DST	DST

References

1. <https://arxiv.org/abs/2002.01359>
2. <https://aclanthology.org/2020.nlp4convai-1.13/>
3. <https://arxiv.org/abs/2010.12757>
4. <https://arxiv.org/abs/2005.00796>