Dialogue State Tracking and Chit Chat Generation

ADL 2021 Final - Team 18

Chun-Hsiang Wang
B06902006
b06902006@csie.ntu.edu.tw

Yun-Ru Lee
B06902107
b06902107@csie.ntu.edu.tw

Bo-Wei Huang
B06902125
b06902125@csie.ntu.edu.tw

Abstract

This report describes our approach in two different NLP missions: Dialogue State Tracking (DST) and Natural Language Generations (NLG).

In our first task, we aim to complete the dialogue state tracking and predict the correct slot value. For the purpose of overcoming two kinds of slots, non-categorical slots and categorical slots, according to our DST mission, we proposed two methods leveraging two end-to-end DST models - Span-DST and Choice-DST respectively to achieve our goal. We use XLNet-Large as our encoder model in Span-DST to predict the value span from the dialogue context, whereas RoBERTa-Large are chosen in Choice-DST to determine whether a specific input slot and value can be inferred from the dialogue context.

In the second task, our goal is to generate chit-chat response in a dialogue with users and systems. We need to determine whether to add a response before or after a system response and generate the appropriate output sentences. In the task of generating chit-chat response, we use T5 as our seq2seq model given the input of all previous dialogue context. Furthermore, we proposed a method to determine whether to add the response using a RoBERTa-based classifier.

Keywords: Dialogue State Tracking, Natural Language Generation

1 Introduction

The goal-driven dialogue system aims to provide a human-computer interaction based on natural language. It has a wide range of applications in intelligent customer services and intelligent personal assistants. A dialogue system is composed of three modules: natural language understanding (NLU), dialogue management (DM) and natural language generations (NLG).

DM is responsible for dialogue tracking and is the core component of goal-driven dialogue system. It controls the continuation of a whole dialogue. A good DM provides a better user experience by enhancing the interaction in a user-system conversation, which can improve the effectiveness of conversation flow. DM contains DST module and action selection module. A dialogue state tracker is usually included in a traditional DM pipeline to monitor the current states of the dialogue system. In this project, we only need to consider all the states contained a dialogue context. Therefore, we proposed an end-to-end method to select the state slots instead of a traditional method which take into account the current utterance and the result of previous DST. Moreover, to handle the unseen task, we use the sentence pair input format to concatenate the whole dialogue and the slot information which contains service and slot description generated by our sentence combining strategy. Also, we proposed two different models to handle the two different situations of non-categorical slots and categorical slots. For the non-categorical slots, we proposed Span-DST to select the value span from the dialogue context. For the categorical-slots, we introduced Choice-DST with two versions to determine if a specific slot value is suitable for out dialogue context. Next, we did some reference alignment which can strongly improve our accuracy in non-categorical slots problem. Last but not least, we proposed a data augmentation method by using back translation to solve the problem of the lack of diversity in the training dataset.

The NLG component in the goal-driven dialogue system accounts for the generation of the system response. It's the most significant part when it comes to the users' impression. The output of the NLG directly puts a great impact on whom the systems interacts with. In this task, our goal is to generate chit-chat response based on the input dialogue context. Adding appropriate chit-chat response in a dialogue could lead to lots of benefits. The humanized utterance makes users enjoy the whole conversations. We divided this problem into two parts. The first part is to generate initial chit-chat response before and after each system response. We use a T5 base seq2seq model to achieve our goal. Furthermore, in order to generate various response instead of monotonous one, we use sampling method to adjust the output. After that, we train a classifier which inputs the dialogue context and response and decide whether to add it. In this section, we build a RoBERTa-based encoder to compute the score between every potential response in each dialogue. Last, we remove some of the repeating response to achieve a more humanizing result.

The project report contains four parts. First in approach section we introduce the model structure and our data augmentation methods. Next in experiment section we list our experiment results and comparison which brings some amazing results with some preprocessing and postprocessing. The

first two parts include DST and NLG. Then is the conclusion. Last is the work distribution in our team.

2 Approach

In this section, we'll introduce the approaches used in Dialogue State Tracking and Natural Language Generation, including but not limited to task formulation, data preprocessing and model structure.

2.1 Dialogue State Tracking

For Dialogue State Tracking, we propose two end-to-end DST models without inheriting information from results of previous turns, which is inspired by [4]. Our method can handle shared information between turns or slots easily and overcome the error accumulation issue in traditional DST systems.

According to the schema, there are two kinds of slots over all services, non-categorical slots and categorical slots. For non-categorical slot, the predicted value is usually filled by the sub-string from the dialogue context, which is essentially an span extraction problem; therefore, we decide to tackle it with Span-DST model. On the other hand, a categorical slot is always equipped with a list of possible values, and its value prediction can be regarded as a multiple-choice problem; thus, we resort to Choice-DST models for help.

Furthermore, due to the lack of service diversity in the training dataset, we utilize back translation on service and slot descriptions for data augmentation. Such extension is aimed to tackle the zero-shot problem, which is one of the main challenges in the competitions.

2.1.1 Span-DST for Non-Categorical Slots.

Given a slot, Span-DST model is supposed to determine the value span from the dialogue context. In Figure 1, we use XLNet-Large proposed in [7] as our encoder model due to its excellent performance in some machine reading comprehension tasks, such as SQuAD 2.0. Also, there are three classifiers on the top of XLNet encoder, i.e. answerable classifier, start position classifier and end position classifier.

Let $S = [s_1, s_2, ..., s_n]$ be a concatenation of all utterances with their own speaker prefix, and $D = [d_1, d_2, ..., d_m]$ is the splicing of service description and slot description, noted as "[service description]: [slot description]". We pack them together with $<cls>$, a special summary token added at the end of every sample, and $</s>$, a special separator token, as the sentence-pair input of XLNet and get their corresponding representations R. Subsequently, an answerable classifier is built upon $<cls>$ encoding and the start position to determine whether the given slot is mentioned in the dialogue by logit Y_{hasAns}. In addition, a start vector V^{start} and an end vector V^{end} are introduced to predict the answer span. We formalize the process as the followings:

$$R = XLNet(S \oplus </s> \oplus D \oplus </s> \oplus <cls>) \quad (1)$$

$$V^{start} = FC_{start}([r_1, r_2, ..., r_n, r_{sep_2}, r_{cls}]) \quad (2)$$

$$V^{end} = FC_{end}([r_1, r_2, ..., r_n, r_{sep_2}, r_{cls}] \oplus r_{startPos}) \quad (3)$$

$$Y_{hasAns} = FC_{ans_2}(tanh(FC_{ans_1}([r_{cls}] \oplus r_{startPos}))) \quad (4)$$

where $FC(\cdot)$ is a dense layer, r_{sep_2} is the representation of the second $</s>$, r_{cls} is the representation of $<cls>$, $r_{startPos}$ is the representation of the first token in answer span.

During training, we apply a logistic regression loss to predict the answerability, along with a standard span extraction loss for span-boundary predictions. Specifically, we calculate the training loss as follows:

$$L_{hasAns} = \sum \log(sigmoid(Y_{hasAns})) \cdot \hat{Y}_{hasAns} \quad (5)$$

$$L_{start} = \sum \log(softmax(V^{start})) \cdot \hat{V}^T_{start} \quad (6)$$

$$L_{end} = \sum \log(softmax(V^{end})) \cdot \hat{V}^T_{end} \quad (7)$$

$$L_{total} = L_{hasAns} + 0.5 \cdot (L_{start} + L_{end}) \quad (8)$$

where $\hat{Y}_{hasAns}$ is the true label of answerability, $\hat{V}_{end}$ is one-hot vector of true start position, and $\hat{V}_{end}$ is one-hot vector of true end position.

During inference, if Y_{hasAns} is smaller than zero, the slot is predicted as not mentioned; otherwise, the following steps are conducted to find the best span boundaries:

1. Calculate V^{start} and keep top 5 as start positions
2. For each start position, calculate V^{end} and keep top 5 as end positions. In total, there are at most 25 start-end combinations kept for further considerations
3. Filter those whose answer span is longer than 30 or start position stays behind end position out of the candidate list
4. Among all, pick the start-end combination with highest sum of start logit and end logit as the final span prediction
5. If $<cls>$ is selected, the slot is predicted as not mentioned. If the second $</s>$ is selected, "dontcare" is the answer

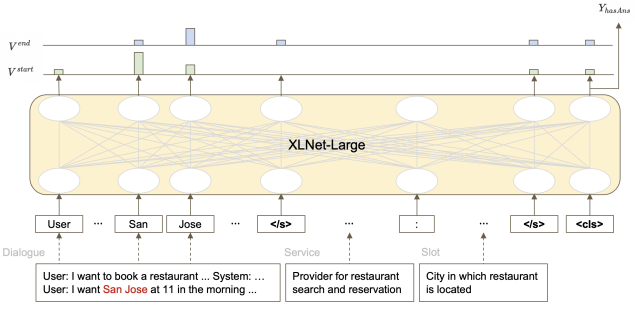

Figure 1. Span-DST model structure

Mid Lang	Text
English	The go-to service for finding and booking appointments with top rated dentists
German	The go-to-service for searching and booking dates with top-rated dentist
Spanish	The service to find and book meetings with high-ranking dentists
Russian	In search and booking orders with the best ranked dentists
Chinese	Find and book dating services with top-rated dentists

Table 1. Back Translation Examples

2.1.2 Choice-DST for Categorical Slots.

Given a slot and a possible value, Choice-DST model is supposed to determine if the slot-value can be inferred from the dialogue context. Note that we insert two additional options, "unknown" and "dontcare", into the list of possible values to simplify the post-processing progress. In Figure 2, we use RoBERTa-Large proposed in [3] as our encoder model with one classifier on the top.

Notably, we consider two versions of classifier inputs. One is deep version, which only contains the representations of encoder. The other one is wide and deep version, which is composed of not only the representations of encoder but also 11 hand-crafted wide features, just like Wide & Deep model mentioned in [4]. The details of these wide features are listed in the Appendix, and the differences in performance between two versions will be further discussed in the Experiments section.

Let $S = [s_1, s_2, ..., s_n]$ be a concatenation of all utterances with their own speaker prefix, and $D = [d_1, d_2, ..., d_m]$ is the splicing of service description, slot description and possible value, noted as "[service description]: [slot description]: [single possible value]". We pack them together with <s>, a special summary token added in the front of every sample, and </s>, a special separator token, as the sentence-pair input of RoBERTa-Large and get the representations of <s> as deep features r_s. Subsequently, in deep version, a classifier is built upon r_s, with 768 dimensions, to decide if the slot-value is a correct statement in the dialogue by $Y_{correct}$; on the other hand, in wide and deep version, a classifier is built upon r_s and 11 additional discrete features, with 779 dimensions in total, to receive the correct logit $Y_{correct}$. We formalize the process as the followings:

$$R = RoBERTa(<s> \oplus S \oplus </s> \oplus D \oplus </s>) \tag{9}$$

$$Y_{correct} = FC_{corr_2}(tanh(FC_{corr_1}(r))) \tag{10}$$

where $FC(\cdot)$ is a dense layer, r is r_s in deep version and $r_s \oplus DiscreteFeatures$ in wide and deep version.

During training, we apply a logistic regression loss to predict the correctness. Specifically, we calculate the training loss as follows:

$$L_{correct} = \sum \log(sigmoid(Y_{correct}) \cdot \hat{Y}_{correct} \tag{11}$$

where $\hat{Y}_{correct}$ is the 0/1 label of slot-value correctness.

During inference, given dialogue and slot, all possible values are first mapped to their corresponding logits. Afterwards, the following steps are implemented to find the most suitable value prediction:

1. Candidate with the highest correct logit becomes the final value
2. If "unknown" is selected, the slot is predicted as not mentioned

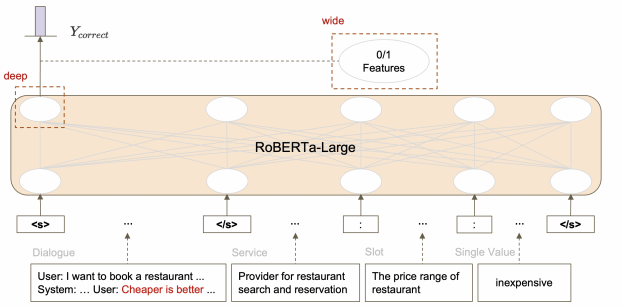

Figure 2. Choice-DST model structure

2.1.3 Back Translation for Data Augmentation.

Through M2M-100 proposed in [1], we translate all descriptions mentioned in schema into a middle language, and then translate it back to English for data augmentation. According to the family group of languages mentioned in [1], we select four languages as middle language candidates, i.e. Spanish, German, Russian and Chinese. Among them, we found that Russian and Chinese can produce more syntactic variations compared to the others, such as table 1, which can positively influence our service and slot diversity.

However, errors always exist. There are some cases that the meanings of back-translated sentences deviate from the original one, just like table 2. Therefore, to keep training stability, we only pick about 60 percent of descriptions in input samples for back translation with randomly selected middle language.

2.2 Natural Language Generation

The goal for natural language generation is to generate chit-chat response before or after the system response. By dividing this problem into two parts can make the final result

Mid Lang	Text
English	Fare per ticket for journey
German	Danger per ticket for the trip
Spanish	Period by ticket for the trip
Russian	Dangerous Tickets for Travel
Chinese	Danger of travel tickets.

Table 2. Back Translation Mistakes

perform well. The purpose of the first part is to generate initial chit-chat response. After that, we make a choice between being chosen or not being chosen for all initial chit-chat response in second part.

2.2.1 Generate chit-chat response.

Since the task in this section can be regarded as one of the summarization task, we use the T5[5] as our seq2seq model. The previous dialogue are the document and the chit-chat response are the summarization labels.

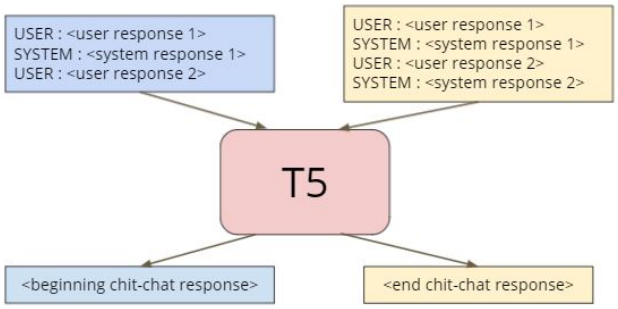

Figure 3. Generate chit-chat model structure

In figure 3, there are two types of the input data in T5. One is to generate beginning chit-chat response which the last dialog is the user utterance; the other is to generate end chit-chat response which the last dialog is the system utterance. The purpose of utilizing the input data is to make the pretrained model can generate the beginning and end chit-chat response at the same time. The functions of two types of input and output are as follows:

$$C_B = T5(U \oplus S \oplus ... \oplus U) \tag{12}$$

$$C_E = T5(U \oplus S \oplus ... \oplus U \oplus S) \tag{13}$$

where C_B is beginning chit-chat response, C_E is end chit-chat response, U is user utterance and S is system utterance.

Besides, in order to make the pretrained model recognize the user and system utterance, we add "USER : " before the all user utterance and add "SYSTEM : " before the all system utterance, respectively.

During fine-tuning, we apply cross entropy loss to compute the difference between model outputs and labels. Therefore, model would like to output the suitable chit-chat response.

After fine-tuning the T5 model, we use different sampling strategies[2] such as beam search and top-k sampling to generate initial chit-chat response. The result of beam search would make the response too generic, so we mix beam search and top-k sampling as our final generation strategy, while the number of beam is 5 and k is 640.

2.2.2 Select chit-chat response.

For improvement of the previous result, we apply a classifier to compute the necessity of all initial chit-chat response. We choose RoBERTa[3] as our encoder model and use a simple classifier for the first token output to compute importance of all potential candidates. Our model structure is shown in figure 4.

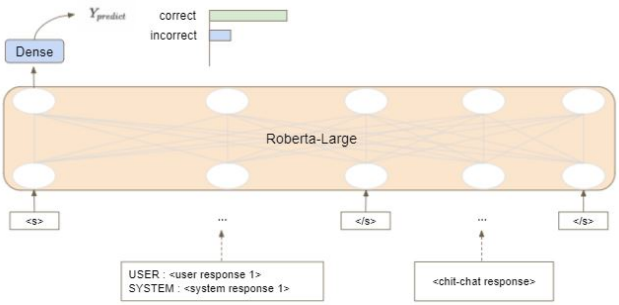

Figure 4. Select chit-chat model structure

The input data contain the dialog and chit-chat response concatenated by $</s>$ that is the eos_token in RoBERTa. We use the first token output as the simple classifier input and reduce the dimension of the first token output to 1 by the simple classifier. Therefore, the dimension of the classifier output is 1, so we can use the output value of it to determine the necessity of the chit-chat response. The functions of RoBERTa and classifier are as follows:

$$O = RoBERTa(<s> \oplus D \oplus </s> \oplus C \oplus </s>) \tag{14}$$

$$Y_{correct} = FC_2(tanh(FC_1(O_0))) \tag{15}$$

where D is the dialog before the chit-chat response, C is the chit-chat response, O_0 is the first token of RoBERTa output, $FC_1(\cdot)$ is a dense layer input and output dimension are same and $FC_2(\cdot)$ is a dense layer output dimension is 1.

On the other hand, we also try the dimension of the classifier output is 2, because the task in this section can be regarded as the binary classification. However, the result of this effort doesn't perform well, so this setting doesn't become our final method.

During fine-tuning, the output would be passed by sigmoid function and compute loss by binary cross entropy that can

	Services	Seen services (Train)	Seen services (Train + Dev)
Dev	20	13	-
Test seen	21	9	13
Test unseen	6	0	0

Table 3. Service analysis of datasets

	Goal Acc	HasAns Acc / Span EM
random ref	0.4927	0.8593 / 0.5709
aligned ref	0.7663	0.8554 / 0.8685
+ data aug	**0.7818**	0.8628 / 0.8754

Table 4. Joint scores of Span-DSTs

make the outputs more likely to predict their true labels of 0/1.

During inference, each chit-chat candidate would be given a value that can determine their importance bypassed the model. After that, we sort all chit-chat candidates by their corresponding values. In addition, by a rule-based method, we remove some common chit-chat responses like `"you're welcome"` because it appears too many times and often finds it in an inappropriate location.

3 Experiments

In this section, we'll show our analysis and experimental results in in Dialogue State Tracking and Chit-Chat Generation.

3.1 Dialogue State Tracking

3.1.1 Datasets and Reference alignment.

When training models, we only utilize the datasets provided in class. Table 3 lists the service analysis of the datasets. It shows that adding development dataset when training can increase seen services for test seen dataset; therefore, in the following experiments, models evaluated on development dataset are trained with train dataset only, while those evaluated on test datasets are trained with train dataset along with development dataset.

Due to some restrictions in Kaggle, only single reference is considered during evaluation. Thus, we encourage our model to align the output answers with TA's arrangements by the following processing:

1. Find the first value that exists in dialogue utterances from the slot-value list
2. If all values in the list are not included in dialogue utterances, we will omit that sample during training, and use the first value in the list as true label during evaluation

Statistically, there are almost 30% of training samples with multiple references for Span-DST, while there are only 6 out of 580,000 training samples with multiple references for Choice-DST. Hence, we assume that such reference alignment might greatly impact the metric scores of Span-DST, and the evidence will be demonstrated afterwards.

3.1.2 Span-DST for Non-Categorical slots.

For Span-DST, we convert single reference into start/end index by using those provided in the datasets. If there are several start/end annotations about the ground-truth value, we'll choose the one closest to the utterance where the slot lastly updates.

We train our models on GeForce RTX 3090 using XLNet-Large pretrained weights from [6] with batch size of 64 and learning rate of 3e-5 for three epochs. The max sequence length is set to 512 and the overflow stride is 256. Also, warm-up scheduler with linear decay is deployed, where the number of warm-up steps is 10% of the whole training steps.

Table 4 shows the performance of Span-DST with different configurations. On the development dataset, reference alignment can significantly improve span exact match by about 30%. We believe that the model learns our reference-processing rules from data to some extent, bringing the span outputs closer to the ground-truth values. Furthermore, data augmentation can improve the performance by about 1.5%, which indicates back-translation can benefit the generalization of Span-DST.

3.1.3 Choice-DST for Categorical slots.

For Choice-DST, we train our models on GeForce RTX 3090 using RoBERTa-Large pretrained weights from [6] with the same hyperparameters as Span-DST, except for the number of epochs. Note that there are multiple possible values for each categorical slots, causing the number of Choice-DST's training samples to be three times more than that of Span-DST's; therefore, we reduce the number of epoch from 3 to 1. Moreover, we notice that there are only 14% of positive examples in this task. To handle such imbalanced dataset, we leverage weighted sampler to increase positive observations during training.

According to schema, all categorical slots can be split into three types: numerical, boolean and text-based slots. The definitions are as follows:

- A numerical slot has a list of all numerical possible values
- Only "True" and "False" are in the list of boolean slot's possible values.
- If a slot is neither numerical nor boolean, then it is a text-based slot

	Goal Acc	Num / Bool / Text Acc
deep	0.8833	0.9631 / 0.9569 / 0.9064
+ data aug	0.8811	0.9646 / 0.9502 / 0.9070
wide and deep	**0.8904**	0.9656 / 0.9712 / 0.9059
+ data aug	0.8900	0.9702 / 0.9635 / 0.9055

Table 5. Joint scores of Choice-DSTs

Span-DST	Choice-DST	Dev	Test seen / unseen
random ref	deep	0.4279	0.3322 / 0.2226
aligned ref	deep	0.6909	0.4206 / 0.1776
+ data aug	+ data aug	0.7003	**0.4586** / 0.2272
aligned ref	wide and deep	0.6950	0.4251 / 0.1689
+ data aug	+ data aug	**0.7073**	0.4488 / **0.2291**

Table 6. Joint Goal Accuracy for overall models

Table 5 shows the joint accuracy along with some detailed performance based on different slot types. On development dataset, Choice-DST with wide and deep features slightly outperform that with deep features by 0.7% in joint accuracy. However, the data augmentation has no improvements on both versions. Also, numerical and boolean slots can always achieve better results than text-based slots, which infers that the possible values of text-based slots are more complicated and uncontrollable, making their predictions tougher than the others'.

3.1.4 Overall.

Table 6 shows the overall joint goal accuracy on several datasets with different Span-DST and Choice-DST combinations, where the scores on test datasets are the means of the public score and private score on Kaggle.

To our surprise, although reference alignment works well on development and test seen dataset, it gets poor performance on unseen services. The possible reason is that such alignment might sacrifice some output diversity and thus hurt the generalization on zero-shot samples. However, the rule says that seen and unseen tasks should share the models. Therefore, we stick to reference alignment because of its significant contributions towards seen services and resort to other configurations for better generalization.

As for the two versions of Choice-DST, generally, the wide and deep one performs better than the deep one. Yet, the differences are not significant enough, so we can hardly determine which one is the winner. The possible causes of impractical wide features are that our hand-crafted features might be not enough, and that the information of our wide features might be already included in the deep ones. Due to time limitation, we leave the modification and expansion of wide features as one of our future works.

Moreover, the data augmentation has some improvements on all datasets, especially on test seen and unseen. Therefore, we can conclude that back translation is an effective tool for data augmentation that can successfully handle the zero-shot cases.

Finally, we choose two model combinations as our final submission in Kaggle. They are (1) Span-DST with reference alignment plus data augmentation and Choice-DST with deep features plus data augmentation, (2) Span-DST with reference alignment plus data augmentation and Choice-DST with deep and wide features plus data augmentation. For private score, the former wins on test seen dataset, while the latter wins on test unseen dataset. If we have to choose one model, we'll pick the first, i.e. one whose Choice-DST uses deep features, because it achieve better average score with less effort than the other.

3.2 Natural Language Generation

In this section, we'll show the details of data preprocessing, experimental settings, result and analysis of chit-chat generation. Our transformers is based on [6] to implement.

3.2.1 Data preprocessing.

For training and development data provided in class, we utilize all chit-chat response to train our propose method. We concatenate the dialog by using some special word. If the chit-chat response is before the system utterance, the dialog data will be "$USER : <userresponse>_1 SYSTEM : <systemresponse>_1 ..... USER : <userresponse>_n$", while if the chit-chat response is after the system utterance, the dialog data will be "$USER : <userresponse>_1 SYSTEM : <systemresponse>_1 ..... USER : <userresponse>_n SYSTEM : <systemresponse>_n$", respectively.

In generation section, we use the chit-chat response whose label is good to train and interface. Otherwise, in selection section, we use all chit-chat response to train and interface.

3.2.2 Generate chit-chat response.

In this section experiment, we use the T5-base as the pretrained language model. We train our model with the initial learning rate of 3e-5 and warm-up scheduler with linear decay is deployed, where the number of warm-up steps is 350. In addition, other training details like batch size is 16 and a maximum sequence length is 512 tokens. Furthermore, the final model parameters selected is the minimum loss when interfacing in the development set.

For chit-chat generation of the test data, we analyze the different sampling strategies like beam search and top-k sampling and an example is shown on table 7. In table 7, the system utterance is *"i have quite a few restaurants, is there is particular type of food you are looking for?"*. Besides, we

Strategy	end chit-chat response
beam search	There are so many to choose from.
+ top-k sampling	I can recommend some places.

Table 7. Compared with different sampling strategies

Output dimension	1	2
Accuracy	0.675	0.734

Table 8. Different output dimension of the classifier result

find that beam search mixed with top-k sampling have more engaging chit-chat response, so we adopt it as our final sampling strategy.

3.2.3 Select chit-chat response.

In this section experiment, we use the RoBERTa-large as the pretrained language model. We train our model with the initial learning rate of 3e-5 and warm-up scheduler with linear decay is deployed, where the number of warm-up steps is 400. In addition, other training details like batch size is 8, number of gradient accumulation steps is 4 and a maximum sequence length is 512 tokens. Furthermore, the final model parameters selected is the maximum accuracy when interfacing in the development set.

In table 8, we compared the accuracy between different output dimensions of the classifier. When the output dimension is 1, we use cross entropy as our loss function. Otherwise, we use binary cross entropy as our loss function when the output dimension is 2. Although the two-dimension output classifier achieves higher accuracy, it's just not good enough to decide what final chit-chat responses are we need. Therefore, we choose the one-dimension output classifier as our final classifier. Though the accuracy of the one-dimension output classifier is lower, it still can use the output value to compute the importance of the chit-chat response.

In the final decision, we sort all chit-chat candidates from high to low by their corresponding value. After that, the first fifty-five percent will be selected for the final chit-chat response. Furthermore, we find some sentences like "`you're welcome`" appearing several times and most of the time that are at an inappropriate location. Thus, we use a rule-based method to remove that kind of unsuitable chit-chat response. Our final chit-chat result is complete by the above methods and experiments eventually.

4 Conclusion

In this final project, we have proposed our methods to solve the DST and chit-chat NLG task. In the DST part, we proposed two end-to-end models with sentence pair form. For non-categorical slots, we trained a Span-DST with XLNet-Large structure. For categorical one we proposed RoBERTa-Large based Choice-DST with two versions of classifier features. We found that reference alignment could improve about 8 percent of the performance on seen services and doing back translation can improve 3 to 5 percent of the total performance. In NLG part, we designed our workflow by firstly generating initial chit-chat response leveraging T5 seq2seq model, then using RoBERTa as the classifier to pick up appropriate response. We compared the result between beam search and top-k sampling in the first part and decided to use top-k sampling due to the lack of diversity of the output response using the first one. We also tried different dimension output of the RoBERTa classifier and found that although using 2 dimension achieves higher accuracy, it's not an appropriate way during inference.

5 Work Distribution

- 王俊翔 B06902006
  - minor DST and minor NLG, report
- 李昀儒 B06902107
  - major NLG, report
- 黃柏瑋 B06902125
  - major DST, report

References

[1] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond English-Centric Multilingual Machine Translation. *CoRR* abs/2010.11125 (2020). arXiv:2010.11125 https://arxiv.org/abs/2010.11125

[2] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The Curious Case of Neural Text Degeneration. *CoRR* abs/1904.09751 (2019). arXiv:1904.09751 http://arxiv.org/abs/1904.09751

[3] 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. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[4] Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. 2019. An End-to-End Dialogue State Tracking System with Machine Reading Comprehension and Wide & Deep Classification. *CoRR* abs/1912.09297 (2019). arXiv:1912.09297 http://arxiv.org/abs/1912.09297

[5] 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. *CoRR* abs/1910.10683 (2019). arXiv:1910.10683 http://arxiv.org/abs/1910.10683

[6] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).

[7] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR* abs/1906.08237 (2019). arXiv:1906.08237 http://arxiv.org/abs/1906.08237

A Appendix

A.1 Wide Features in Choice-DST

There are 11 discrete features mentioned in Choice-DST (wide and deep). Both of them are in one-hot format. Given one dialogue, service, slot and value, the construction of features are mainly covered by the following aspects:

- The dataset source is SGD or MultiWOZ
- Whether the dialogue contains multiple services
- The slot type is numerical, boolean or text. The definition of slot types are mentioned in section 3.1.3.
- Whether the value is "unknown" or "dontcare"
- Whether the utterances contain slot words or their synonyms
- Whether the utterances contain value or their synonyms if slot type is numerical or text
- Whether contexts around slot words contain negative words if slot type is boolean

Note that the slot words are nouns, adjectives or past participles of verb in slot name. Also, when it comes to synonyms, we convert numerical values directly to words, and use WordNet for other slot types' values.