Adapting Pre-trained Language Model for Dialogue State Tracking on Spoken Conversations

Haeun Yu,[1] Taesuk Hong,[2] Youngjoong Ko,[3*] Kwangyong Lee[4]

Sungkyunkwan University,[1, 3] Sogang University,[2] LG Electronics[4]
haeun.yu204@gmail.com,[1] lino.taesuk@gmail.com,[2] yjko@skku.edu,[3] kwangyong19.lee@lge.com[4]

Abstract

The 10th Dialogue State Tracking Challenge (DSTC10) Track 2 released a new task-oriented dialogue dataset on spoken conversations for dialogue state tracking. This dataset has some distinct characteristics that are different from those of the existing task-oriented dialogue dataset. In this paper, we present adaptation approaches from the model and data perspectives for the dataset released by DSTC10 Track 2. For the approaches from the model perspective, we add an additional slot type "multiple" and a convolutional layer for a better performance. For the approach from a data perspective, we collected and augmented the dataset. We also devised a new post-processing approach to deal with errors in spoken conversations. Using our approaches, we improved the performance of the model based on the DSTC10 official baseline from 0.3% absolute points to 18% absolute points. Consequently, our model ranked fifth in DSTC10 Track 2 Task 1.

1 Introduction

Dialogue state tracking (DST) in a task-oriented dialogue system is a task that aims to track users' needs appearing in dialogue to achieve the user's goal successfully. Specifically, it can be categorized as slot-filling task. A slot refers to the category of the user's need, and each slot has its corresponding value. In the 10th Dialogue System Track Challenge (DSTC10), a new DST dataset built from spoken conversations was released for Task 1 of Track 2.

In this paper, we describe how to adapt the DST model to the dataset released by DSTC10 Track 2. Track 2 does not provide any training data and only releases the development set for the competition. We refer to this development set as the DSTC10 development set for convenience. The development set released by DSTC10 Track 2 has two main problems that need to be addressed. Firstly, several slots, such as "hotel-stars" and "restaurant-time," can be filled with more than one value. A slot will contain more than one value when the user does not mention the specific preference regarding the slot. For example, "at least a 2-star hotel" rather than a "2-star hotel." In this case, the value for "hotel-stars" will be "2, 3, 4, 5 stars." We call these values multiple values. Multiple values do not appear in existing dialogue datasets and cannot be handled by current DST models. Secondly, the dataset contains three times more entities than the well-known task-oriented dialogue dataset, MultiWOZ 2.1. As Track 2 does not release training data, new entities and their information, such as area and food, that appear in the DSTC10 development set require an approach that aims to adapt the model trained on the existing dialogue dataset to a new dataset. Lastly, as the dataset is built from human-human spoken conversations, it contains the n-best automatic speech recognition (ASR) hypotheses for utterances. There is also the need for an approach to tackle the errors produced by the ASR outputs.

To adapt the model, we improved the existing model from two perspectives: data and model. To overcome the dataset-related problem, we collected and augmented the training data for the DSTC10 development set in line with MultiWOZ 2.1. In addition, we modified the existing state-of-the-art DST model and developed post-processing approaches to adapt the model effectively to new dataset.

Our contributions can be summarized as follows:

- From the model perspective, the convolutional layer was added to obtain a better span prediction performance, and a new class type called "multiple" was added as an adaptation approach for tracking multiple values.
- From the data perspective, the training data were collected and augmented to adapt the existing model to a new dataset.
- Considering the characteristics of the task-oriented dialogue, a post-processing method was devised to capture the missed values during state tracking.

2 Related Works

DST can be solved in various ways, such as ontology-based classification, value generation, and value span prediction. Traditionally, the values of the dialogue state slot have been filled by classification among the predefined set of values for each slot. However, as the need to understand longer, more complex dialogue arises (Budzianowski et al. 2018), this classification approach is no longer considered as a competitive method. With the advance of pre-trained language models such as BERT (Devlin et al. 2019), and GPT2 (Radford et al. 2019), the value generation, and value span prediction approaches have emerged as an alternative to traditional methods in a newly arisen open-vocabulary DST set-

*Corresponding author

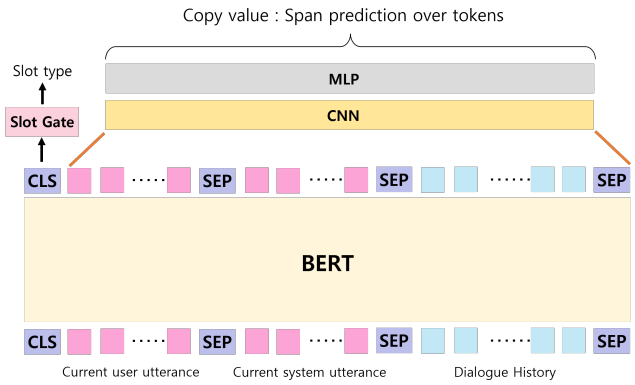

Figure 1: Overview of the model

.

ting. SimpleTOD (Hosseini-Asl et al. 2020) tracks the value of slots by generating slots and values sequentially given the dialogue history. It is trained with a simple language modelling objective. TripPy (Heck et al. 2020) is a DST model with a span prediction approach, and recently proposed DST models are built upon this model. It conducts classification on the CLS token to predict the type of slot and span prediction to extract values from the dialogue utterances. The current state-of-the-art DST model on the MultiWOZ 2.1 dataset is SaCLog (Dai et al. 2021) which integrates curriculum learning with TripPy. Using the same model, SCoRe (Yu et al. 2021) implements an additional training approach before fine-tuning, to model the relationship between the utterances and the structured database. Moreover, CoCo (LI et al. 2021) showed comparable results to the current state-of-the-art model with a new data augmentation approach.

3 Methods

In this section, approaches from model and data perspectives to adapt the model to a new dataset are described. Firstly, the approaches from the model perspectives are discussed. And then, we describe how we collected and augmented the data. Lastly, we also describe our additional approaches to obtain a better performance, post-processing and ensemble methods.

3.1 Approaches from Model Perspective

From the model perspective, we modify the TripPy model in two ways. Figure 1 presents an overview of the modified model. The first approach involves handling multiple values. Through the analysis of the DSTC10 development set, we observed some common situations that always had multiple values. For instance, when the booking of a restaurant was not successful for the time first mentioned by the user, the user revised the booking with two possible times, before or after one hour from the time mentioned previously. In this case, the multiple values of the "restaurant-time" become "12:00" and "14:00," if the user first intented to book for "13:00." We organize these situations as patterns and create four patterns for four slots: "hotel-day," "hotel-stars," "restaurant-pricerange," and "restaurant-time."

Original dialogue from MultiWOZ2.1 (SNG1006)	
User: Okay, I'd like to book a room at the Gonville Hotel for 4 nights. There will be 6 people and we will be arriving on Saturday.	
System: I'm sorry, there are no rooms available for that length of stay. Could you shorten your stay or book a different day possibly?	hotel-day: ["saturday"]
User: Yes, what about 2 nights instead of 4?	
System: Sure, that worked. You have booked 2 nights and your reference number is ru89u6v8. Can I be of further help today?	
Modified dialogue for Multi-value tracking	
User: Okay, I'd like to book a room at the Gonville Hotel for 4 nights. There will be 6 people and we will be arriving on Saturday.	
System: I'm sorry, there are no rooms available for that length of stay. Could you shorten your stay or book a different day possibly?	hotel-day: ["saturday"]
User: Yes, let's check if they have any day either this Friday before or Sunday after.	
System: Sure, that worked. You have booked 2 nights and your reference number is ru89u6v8. Can I be of further help today?	hotel-day: ["friday", "sunday"]

Figure 2: Example of the collected dialogue for multiple values

.

To utilize these patterns, we add "multiple" type to the slot type of TripPy, which determines the way to fill the value of the slot. Originally, the model contained seven different types of slots: none, dontcare, span, inform, refer, true, and false. In addition to adding a new slot type, we also deleted the true and false slot types because they are intended for "hotel-internet" and "hotel-parking" slots, which do not appear in the DSTC10 development set. Multiple values are filled when the slot type is classified as "multiple" following the patterns constructed through the analysis of the DSTC10 development set.

Our second approach is to enhance the performance of span prediction in dialogue state value detection. Values of each slot of dialogue state are often detected at the similar position of given dialogue context. Thus, learning positional information can help in detecting the value for the dialogue state. To obtain the positional information, we added a convolutional layer between the BERT model and the multilayer perceptron (MLP) layer. As the convolution layer takes the neighboring tokens of a certain token to produce the embedding of that token, we can produce an embedding for the token that contains the information of the neighboring tokens. With this convolutional layer, the model can learn the general context of the span of the value that needs to be tracked.

3.2 Approaches from Data Perspective

Data Collection for Multiple Values As no training data examples are available for tracking multiple values, we hired three people to modify dialogues in MultiWOZ 2.1 to obtain training examples that contain multiple values for slots. We use the patterns of the multiple values described in Section 3.1 and modify the dialogues to follow the patterns. Con-

Original dialogue from MultiWOZ 2.1 (SNG0666)	
User: I'd looking for a good restaurant on the east side. I'd prefer the moderate price range, if possible. (Delex) I'd looking for a good restaurant on <restaurant-area>. I'd prefer the <restaurant-pricerange> price range, if possible.	
System: Would you prefer Indian or Italian food? (Delex) Would you prefer <restaurant-food> or Italian food?	restaurant-area: ["east"], restaurant-pricerange: ["moderate"]
User: I would like Indian please. (Delex) I would like <restaurant-food> please.	
System: There is curry prince, and rajmahal. Would either of those work for you? (Delex) There is <restaurant-name>, and rajmahal. Would either of those work for you?	restaurant-food: ["Indian"]
User: I'd like to book curry prince for 6 people at 15:45 on Saturday please. (Delex) I'd like to book <restaurant-name> for 6 people at 15:45 on Saturday please.	
System: Alright, I have you reservation made. The reference number is smamrcq9.	restaurant-name: ["curry prince"] restaurant-people: ["6"] restaurant-time: ["15:45"] restaurant-day: ["Saturday"]
New Entity Information name: Royal Indian Cuisine on Fillmore / food: Himalayan/Nepalese / type: restaurant / address: 1740 Fillmore St / postcode: 94115 / area: Lower Pacific Heights / city: San Francisco / phone: (415) 567-7789 / pricerange: moderate	
Augmented dialogue with new entity	
User: I'd looking for a good restaurant on Lower Pacific Heights. I'd prefer the moderate price range, if possible.	
System: Would you prefer Himalayan/Nepalese or Italian food?	restaurant-area: ["lower pacific heights"], restaurant-pricerange: ["moderate"]
User: I would like Himalayan/Nepalese please.	
System: There is Royal Indian Cuisine on Fillmore, and raajmahal. Would either of those work for you?	restaurant-food: ["Himalayan/Nepalese"]
User: I'd like to book Royal Indian Cuisine on Fillmore for 6 people at 15:45 on Saturday please.	
System: Alright, I have you reservation made. The reference number is smamrcq9.	restaurant-name: ["Royal Indian Cuisine on Fillmore"] restaurant-people: ["6"] restaurant-time: ["15:45"] restaurant-day: ["Saturday"]

Figure 3: Example of the augmented dialogue for entity adaptation

.

sequently, 1300 dialogues for four types of slots are generated in total. An example of the collected data is presented in Figure 2. This example shows a dialogue that contains multiple values for "hotel-day" slot. We change the original utterance of the user to express the user's suggestion to check the availability for the previous or next day. The value for "hotel-day" slot becomes a single value if the final booked day is mentioned explicitly; otherwise, it remains as multiple values. We up-sample the collected dialogues to handle the imbalance between the number of dialogues that contain multiple values and the number of dialogues that do not contain multiple values. For dialogues that are modified to contain multiple values for "hotel-stars" and "restaurant-pricerange," we up-sample the number of dialogues three times. For dialogues that are modified to contain multiple values for "hotel-day" and "restaurant-time," we up-sample the number of dialogues twice.

Data Augmentation using Entity Replacement Some entities in the DSTC10 database do not appear in the training data. To ensure that the model trains each entity at least 10 times in the training phase, we augment the dialogues with entities in the DSTC10 database. We first create the template by delexicalizing existing dialogues. To delexicalize, we replace the value span of entity-related slots, such as "restaurant-food," "hotel-area," and "attraction name," with a special "domain-slot" token. We obtain 100 different templates via delexicalization. We then replace the delexicalized tokens with the corresponding values of the entity. Figure 3 illustrates an example of an augmented dialogue. (Delex) in Figure 3 indicates delexicalized utterances. The original entity name "Curry Prince" and the entity-related information appearing in the dialogue are replaced with the new entity "Royal Indian Cuisine on Fillmore" and its information.

3.3 Post-processing

In a task-oriented dialogue, the system recommends one or more entities to the user that match the conditions given through the conversation. For example, consider that a user wants to find a restaurant according to the following conditions: area, food, and pricerange. These conditions are usually mentioned first, and then the system presents the user the right entity matching these conditions. Regarding this characteristic, we find a pattern that the "domain-name" slot is determined after the conditional slots are filled, e.g., "restaurant-food," "hotel-pricerange," and "attraction-type." Thus, when the prediction of the model includes the "domain-name" slot in the current turn, other conditional slots can be filled by the corresponding information listed in the database (DB). We refer to this method as DB-grounded post-processing. Figure 4 shows the flow of the DB-grounded post-processing.

In addition to DB-grounded post-processing, we create two types of dictionaries to handle the errors in spoken conversations. The first is to correct the spelling error of the value produced by ASR. The key of this dictionary is the entity span extracted from the dialogue containing the spelling error, and the value is its corrected one. The other dictionary is called the headword dictionary. This dictionary is used to handle the abbreviation of entity names in spoken dialogue. For example, in spoken dialogue, instead of speaking the full entity name, people use its abbreviated version, e.g., they say "Plaj" instead of "Plaj Scandinavian restaurant & bar." The key of this dictionary is one or two words of the entity name, and the value is the full entity name.

3.4 Ensemble

For the ensemble, we train five BERT-base models and five RoBERTa-base models with five different seeds, and we at-

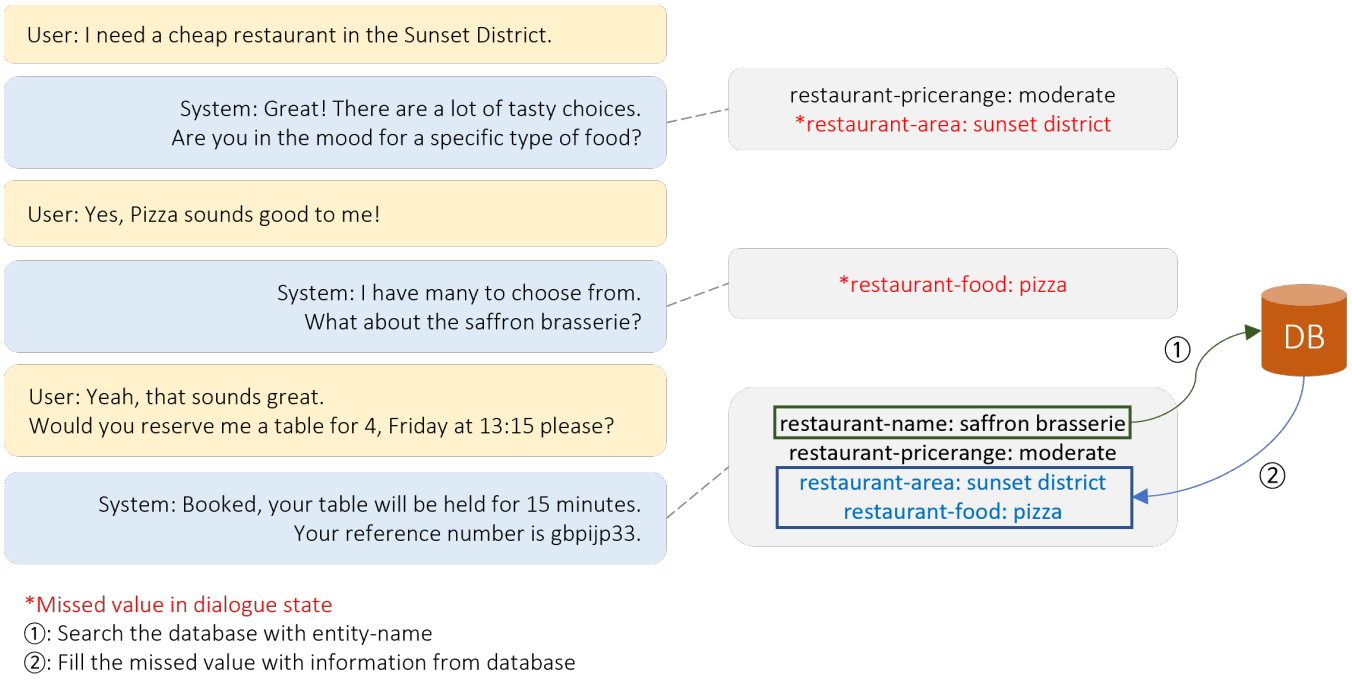

Figure 4: Flow of DB-grounded post-processing

.

tempt two different ensemble methods. The first method is the combination of the best slot accuracy from the trained models, called as ensemble-combination later in this paper. We measure the prediction accuracy of each model for each slot for the DSTC10 development set. We then assign the highest scoring model to each slot. For the prediction of the test dataset from DSTC10 Track 2, we determined the value for each slot using each assigned model. The second is the voting of the predictions from the five best joint goal accuracy (JGA) models. We refer to this method as ensemble-voting. Instead of measuring the prediction accuracy of the models for each slot, we only consider JGA of each model and select top five models according to the JGA on the DSTC10 development set. From the model predictions, we select the most common value as the final value for each slot. In case of the "none" value, even if there exists a value, if the "none" value is the most common value among the five predictions, we select "none" as our final prediction for that slot. The results of each ensemble method are discussed in the following section.

4 Experiments and Results

In this section, the experimental setup and the results of our method are discussed. We report all the performances on the DSTC10 test set.

4.1 Datasets

As no official training data were provided, we used the MultiWOZ 2.1 train dataset as our main training dataset. In addition, we collected and augmented the data according to the methods described above. The statistics of the training dataset are shown in Table 1. Although the development set of DSTC10 Track 2 (Kim et al. 2021) only contains slots from the hotel, restaurant, and attraction domain, we included slots from train and taxi domains appearing in the MultiWOZ 2.1 dataset to preserve the good performance of the original model. During the evaluation using the DSTC10 development set, we only use 1-best ASR output among n-best ASR outputs.

Number of MultiWOZ 2.1 train sets	8438
Number of collected multiple value dialogues	1372
Number of augmented dialogues	8550

Table 1: Statistics of training data including collected and augmented data

4.2 Training Details

We trained our model using the PyTorch framework (Paszke et al. 2019). We implemented our approach on BERT-base and RoBERTa-base. Following SaCLog (Dai et al. 2021)'s approach, we conducted an auxiliary pre-training of the pre-trained transformer encoder model before the fine-tuning stage. And then, we trained the model for 30 epochs at the fine-tuning stage. For the auxiliary pre-training, we used Masked Language Modeling (MLM) objective with the MultiWOZ 2.1 dataset and DSTC9 Track 1 knowledge-grounded dialogue dataset. The model was trained for five epochs using the AdamW optimizer and the learning rate of 1e-5. For fine-tuning, the transformer encoder was only

	JGA	Slot Accuracy	Value Precision	Value Recall	Value F1-score	None Precision	None Recall	None F1-score
DSTC10 Official Baseline	0.39	70.52	61.30	30.97	41.15	73.02	97.16	83.38
Single model	16.97	86.81	84.45	72.63	78.09	88.32	96.58	92.27
Ensemble - Combination	17.32	87.2	85.69	72.74	78.68	88.32	96.58	92.27
Ensemble - Voting	18.21	87.59	87.67	73.32	79.86	87.78	97.38	92.33

Table 2: Results of our experiments with two ensemble methods

	JGA	Slot Accuracy	Value Precision	Value Recall	Value F1-score	None Precision	None Recall	None F1-score
(a) *	4.11	75.41	63.34	43.53	51.6	80.27	97.05	87.87
(a) + (b) *	6.69	77.71	60.13	52.4	56	87.44	94.96	91.05
(a) + (b)	8.24	80.58	67.69	57.69	62.29	87.51	96.15	91.62
(a) + (b) + (c)	8.4	80.36	67.81	56.37	61.56	86.82	96.62	91.46
(a) + (b) + (c) + (d)	16.97	86.81	84.45	72.63	78.09	88.32	96.58	92.27

Table 3: Results of the ablation study of our approaches. The model was trained for 30 epochs by default. For * marked results, the model was trained for 10 epochs.

Rank	Team ID	JGA
1	A11	46.16
2	A01	36.05
3	A07	27.73
4	A10	26.79
5	**A09(ours)**	**18.21**
6	A06	16.91
7	A05	16.15
8	A03	05.24
9	A04	00.50
10	A08	00.18
11	A02	00.14
Baseline		00.39

Table 4: Overall results of the automatic evaluation of DSTC10 Track 2 Task 1

initialized with the post-trained weights, and the other layers, such as MLP, CNN, are randomly initialized. We set the filter size of the convolutional layer to 3. Sequences longer than 512 were truncated. We also set the learning rate to 1e-4, and used a single GPU with 16 batch sizes, two accumulation steps with the AdamW optimizer. For the other hyper-parameters, we used the default hyper-parameter setting of the TripPy (Heck et al. 2020).

4.3 Evaluation Metrics

The most important evaluation metric for DST is JGA. It measures the performance at the turn-level, where the prediction of the model is regarded as accurate if all the slots and values in the prediction are correct. This metric is used as the main evaluation metric in DSTC10 Track 2 Task 1. As the JGA is strict, the slot-level accuracy is also considered. Precision, recall and F1-score are also used to evaluate the performance of the model.

4.4 Experimental Results

Table 2 shows the results of our three submissions compared with the DSTC10 official baseline. The baseline model from DSTC10 Track 2 is widely used for the MultiWOZ 2.1 dataset, TripPy. Compared with the official baseline, our approaches improved the overall performance. Ensemble-voting demonstrated the best performance outperforming the baseline with 17.82%, 17.07% absolute point improvements in JGA and slot accuracy on the DSTC10 test set, respectively. For ensemble-voting and ensemble-combination, two RoBERTa-base models and three BERT-base models were selected. Although the difference of slot accuracy between two ensemble models was about 0.4%, this difference accounts for about 1% of the difference in terms of JGA. In addition, the overall results of the automatic evaluation of the DSTC10 Track 2 Task 1 are shown in Table 4. Our team, A09, ranked fifth among eleven entries.

4.5 Ablation Study

We also conducted an ablation study to determine the effectiveness of our approach. Our base model architecture is a modified TripPy with an additional slot type, namely "multiple," and an additional convolutional layer, as mentioned in Section 3.1. All the results in table 3 are on the test set released at the end of the challenge. Our setup for the ablation study was as follows:

(a) Add collected data for multiple values to the training data.

(b) Add augmented data for new entities to the training data.

(c) Conduct an auxiliary pre-training of the transformer encoder model with MLM objective before fine-tuning step.

(d) Perform post-processing.

The results of the ablation study showed that all our approaches performed well. From Table 3, we can observe that the JGA of the modified model is better than that of the official DSTC10 baseline model by 3.72% absolute points. In addition, we observe that our post-processing approach is the most effective approach. When the post-processing was performed, the JGA score was improved over that of the (a) + (b) + (c) model by 8.57% absolute points.

5 Conclusion

In this paper, we described our approaches for adapting the DST model to the dataset from DSTC10 Track 2. Our ap-

proaches contain two perspectives: model and data. We enhanced the performance of the existing DST model with an additional slot type and convolutional layer. In addition, we collected and augmented the dialogue examples using the MultiWOZ 2.1 dataset. The results showed that our approach effectively improves the performance of the model. In the future, we will investigate ways to handle the errors from ASR output at the model-level, reducing the manual post-processing.

Acknowledgements

This work was supported in part by LG Electronics (CTO/AI lab) and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2020R1A2C2100362).

References

Budzianowski, P.; Wen, T.-H.; Tseng, B.-H.; Casanueva, I.; Stefan, U.; Osman, R.; and Gašić, M. 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 (EMNLP)*.

Dai, Y.; Li, H.; Li, Y.; Sun, J.; Huang, F.; Si, L.; and Zhu, X. 2021. Preview, Attend and Review: Schema-Aware Curriculum Learning for Multi-Domain Dialogue State Tracking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 879–885. Online: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Heck, M.; van Niekerk, C.; Lubis, N.; Geishauser, C.; Lin, H.-C.; Moresi, M.; and Gasic, M. 2020. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 35–44. 1st virtual meeting: Association for Computational Linguistics.

Hosseini-Asl, E.; McCann, B.; Wu, C.-S.; Yavuz, S.; and Socher, R. 2020. A Simple Language Model for Task-Oriented Dialogue. arXiv:2005.00796.

Kim, S.; Liu, Y.; Jin, D.; Papangelis, A.; Gopalakrishnan, K.; Hedayatnia, B.; and Hakkani-Tur, D. 2021. "How robust r u?": Evaluating Task-Oriented Dialogue Systems on Spoken Conversations. arXiv:2109.13489.

LI, S.; Yavuz, S.; Hashimoto, K.; Li, J.; Niu, T.; Rajani, N.; Yan, X.; Zhou, Y.; and Xiong, C. 2021. CoCo: Controllable Counterfactuals for Evaluating Dialogue State Trackers. In *International Conference on Learning Representations*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

Yu, T.; Zhang, R.; Polozov, A.; Meek, C.; and Awadallah, A. H. 2021. SCoRe: Pre-Training for Context Representation in Conversational Semantic Parsing. In *International Conference on Learning Representations*.