DOMAIN-SLOT RELATIONSHIP MODELING USING A PRE-TRAINED LANGUAGE ENCODER FOR MULTI-DOMAIN DIALOG STATE TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dialog state tracking for multi-domain dialogs are challenging because the model should be able to track dialog states across multiple domains and slots. Previous studies had two drawbacks: First, the relationship among different domain-slot pairs were not modeled, restricting the model's capability. Second, models processed each domain-slot pair individually, costing much computation. To tackle these issues, we propose a model for multi-domain dialog state tracking that effectively models the relationship among domain-slot pairs using a pre-trained language encoder. Inspired by the way the special $[CLS]$ token in BERT is used to aggregate the information of the whole sequence, we used multiple special tokens for each domain-slot pair that encodes information corresponding to its domain and slot. We introduce two layouts of such special tokens, *DS-merge* and *DS-split*, and compare their performance. Theses special tokens are run together with the dialog context through the pre-trained language encoder, which enables modeling the relationship among different domain-slot pairs. Our approach computationally efficient since it does not run multiple times as the number of domain-slot pairs. The experimental results show that our model achieves state-of-the-art performance on the MultiWOZ-2.1 Dataset among models without any extra-supervision.

1 INTRODUCTION

Dialog State Tracking (DST) is one of the core components of task-oriented dialog systems. Task-oriented dialog system helps humans get the information that they seek to find for a certain tasks such as finding where an attraction is or reserving a restaurant by understanding the need of the human and providing relevant information. As a component of the task-oriented dialog system, DST aims to identify the state of the dialog between a user and the system, which is represented as {*domain, slot, value*} triplets. Examples of such pairs are {*restaurant, price range, cheap*} and {*train, arrive-by, 7:00 pm*}. Fig. 1 gives an example of how the dialog state changes during the conversation between the user and the system. Since a dialog continues for multiple turns of utterances, the DST model should successfully predict the dialog state as the conversation proceeds. For multi-domain conversations, the DST model should be able to track dialog states across multiple domains and slots and acknowledge when the domain of the conversation changes and predict the dialog state accordingly.

Previous research on multi-domain conversations used a placeholder in the model for representing each domain-slot pair and ran the model multiple times by inserting each domain-slot pair into the placeholder (Wu et al., 2019; Zhang et al., 2019; Lee et al., 2019). A typical DST model for multi-domain dialog uses an input of dialog context until turn t, X_t and domain-slot pair information DS_i for domain-slot i and runs it through an encoder: $f_{model}(X_t, DS_i)$ for $i = 1, \cdots, N$, where N is the number of domain-slot pairs. X_t is composed of a sequence of user and system utterances and DS_i are represented as embeddings. However, this approach has two drawbacks. First, because each domain-slot pair is modeled independently, the relationship among the domain-slot pairs can not be modeled. For example, if the user first asked for a hotel in a certain place and later asked for a restaurant near that hotel, sharing the information between *hotel-area* and *restaurant-area* would help the model recognize that the restaurant should be in the same area as the hotel. Second,

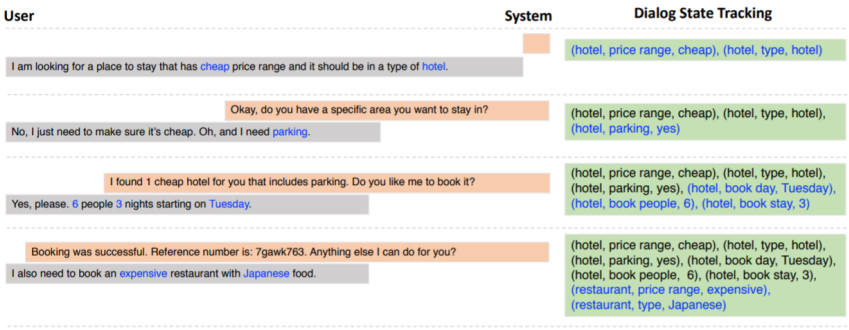

Figure 1: An example of dialog state tracking process. Figure from (Zhang et al., 2019)

because the model has to run multiple times for each domain-slot pair, the computation cost for training and inference increases rapidly. For example, models that encode the domain-slot pair information with the dialog context using the Transformer (Vaswani et al., 2017), the computation cost is $O(k \times (n+1)^2)$, where n is the number of tokens and k is the number of domain-slot pair.

To tackle these problems, we propose a model that can detect the dialog state of multiple domain-slot pairs in a single run of the model given a dialog history, which also leads to modeling the relationship among the domain-slot pairs. Our approach can be represented as: $f_{model}(X_t, DS_1, \cdots, DS_N)$. Because all the domain-slot pair information is run together through the model, interactions among different domain-slot pairs are possible. Moreover, if we assume the same Transformer model mentioned above, the computation cost is $O(n+k)^2$, which is much less than $O(k \times (n+1)^2)$, since $n << k$ in most cases. Also, with the advent of BERT (Devlin et al., 2019), various research incorporated BERT-style encoders for encoding the dialog history (Zhang et al., 2019; Lee et al., 2019; Chao & Lane, 2019; Gao et al., 2019). In this light, we used ALBERT (Lan et al., 2019) as our encoder, which models the relationship among domain-slot pairs with respect to the dialog context. We chose ALBERT among many BERT-style encoders because of its strong performance on numerous natural language understanding tasks while having fewer parameters compared to other BERT-style encoders.

The $[CLS]$ token of BERT (Devlin et al., 2019) is expected to encode the aggregate sequence representation as it runs through BERT, which is used for various downstream tasks such as sentence classification or question answering. This $[CLS]$ token can also be used as an aggregate representation for a given dialog context. However, in a multi-domain dialog, we thought it would be difficult for a single $[CLS]$ token to store information for different domain-slot pairs at the same time. In this light, we propose to use multiple special tokens, one for each domain-slot pair. Each special token works like the $[CLS]$ token of BERT, but in a way that aggregates information that corresponds its domain-slot pair. We represent such tokens in two different ways. The first method, which we call *DS-merge*, we use a single token to represent a single domain-slot pair. For example, to represent a domain-slot pair of *restaurant-area*, we use a special token $DS_{\text{restaurant-area}}$. The second method, which we call *DS-split*, we use separate tokens for the domain and slot, and merge them to represent a domain-slot pair. For the same running example, we merge the domain token $D_{restaurant}$ and the slot token S_{area}. We will use $\{DS\}_{merge}$ and $\{DS\}_{split}$ for the special tokens for *DS-merge* or *DS-split*, respectively. $\{DS\}$ will be use to represent both methods collectively. The $\{DS\}$ tokens, after being encoded by ALBERT along with the dialog context, is used to predict its corresponding domain-slot value for a given dialog context.

2 Related Works

Recent work on dialog state tracking can be largely divided into two groups according to how the slot-values are predicted: fixed-vocabulary and open-vocabulary. The fixed-vocabulary approach, also known as the picklisted-based approach, uses a classification module to predict the dialog state for each slot from a pre-defined set of candidate values (Zhong et al., 2018; Nouri & Hosseini-Asl, 2018; Ramadan et al., 2018; Eric et al., 2019; Lee et al., 2019; Chen et al., 2020). The open-

vocabulary approach generates the dialog state for each domain-slot either by using a generative decoder to generate text (Wu et al., 2019; Hosseini-Asl et al., 2020) or by extracting text spans from the dialog history (Gao et al., 2019; Goel et al., 2019; Heck et al., 2020). There is also an approach to use both picklist-based and span-based methods according to the slot type (Zhang et al., 2019).

For models that deal with multi-domain dialog, how they deal with different domain-slot pairs is another way to divide them. The first approach encodes the dialog context indepedent of the domain-slot pairs and uses separate modules for each domain-slot pair (Eric et al., 2019; Gao et al., 2019; Goel et al., 2019; Heck et al., 2020). The second approach encodes the dialog context using the domain-slot pair information as the prefix and run the encoder multiple times (Nouri & Hosseini-Asl, 2018), (Wu et al., 2019). Also there is another apporach which encodes the dialog context independently but merges it with domain-slot pair information later with a separate fusion module (Zhong et al., 2018; Ramadan et al., 2018; Lee et al., 2019). However, none of these models are able to model the relationship among different domain-slot pairs because there no such module that enables the interaction between them.

(Hosseini-Asl et al., 2020) and (Chen et al., 2020) directly or indirectly involves modeling the relationship among different domain-slot pairs. (Hosseini-Asl et al., 2020) form a model of multi-task learning that encompassess DST as well as action and response generation with a generative language model GPT-2 (Radford et al., 2019). In the decoding process, different domain-slot pairs are generated and this enables indirect interaction among them. However, since the model is trained in a multi-task learning framework, it uses additional labels such as system action durinng training, whereas our model only uses the dialog state label and dialog context. (Chen et al., 2020) explicitly models the relationship among different domain-slot pairs by using a Graph Attention Network (GAT) (Veličković et al., 2018). Schema graphs, which is the relation graph between domains and slots are also used. Our work is different in that we leverage ALBERT, a pre-trained language encoder instead of a GAT. We also do not use pre-defined relation graphs between different domain and slots.

3 PROPOSED METHOD

Our model is composed of three parts. The first is the domain-slot-context (DSC) encoder, which encodes the dialog context along with the special tokens representing domain-slot pairs. Next is slot-gate classifier, which is a preliminary classifier that predicts whether each domain-slot pair appears on the dialog context. The last is the slot value classifier for predicting the candidate values for each domain-slot pair. In the following descriptions, we assume a dialog context with a total of T turns. The task is to predict the dialog state, which is {*domain, slot, value*} triplets for all domain-slot pairs, for every turn $t = 1, \cdots, T$, using the dialog context until each turn.

3.1 DOMAIN-SLOT-CONTEXT ENCODER

The main structure of our model is the DSC encoder, which uses ALBERT to encode the dialog context along with $\{DS\}$ tokens. $\{DS\}$ tokens are expected to work as the $[CLS]$ token for BERT, encoding information corresponding to its domain-slot pair (*DS-merge*) or domain and slot (*DS-split*). The set of special tokens for each layout are shown in Eq. (1) and Eq. (2), respectively. In *DS-merge*, we used special tokens for each individual domain-slot pair. For example, to represent a domain-slot pair of *restaurant-area*, we use a special token $DS_{restaurant-area}$. If there are many domain-slot pairs, using this layout can increase the number of special tokens as each domain-slot pair requires a separate special token. In *DS-split*, we used separate tokens for the domain and slot. To represent a domain-slot pair, we merged the corresponding tokens from each domain and slot by concatenating them. To represent a domain-slot pair *restaurant-area*, we use the domain token $D_{restaurant}$ and the slot token S_{area}. This promotes modeling compositionality, since the same slot token can be used for different domains. These $\{DS\}$ tokens, which is essentially domain and slot specific $[CLS]$ tokens, are used in coordination with the dialog context to encode contextualized representations according to what each token represents.

$$\{DS\}_{merge} = \{\mathrm{DS}_{\mathrm{domain_{(i)}\text{-}slot_{(j)}}}, \cdots, \mathrm{DS}_{\mathrm{domain_{(n)}\text{-}slot_{(m)}}}\} \tag{1}$$

$$\{DS\}_{split} = \{D_{domain_{(i)}}, \cdots, D_{domain_{(n)}}, S_{slot_{(j)}}, \cdots, S_{slot_{(m)}}\} \tag{2}$$

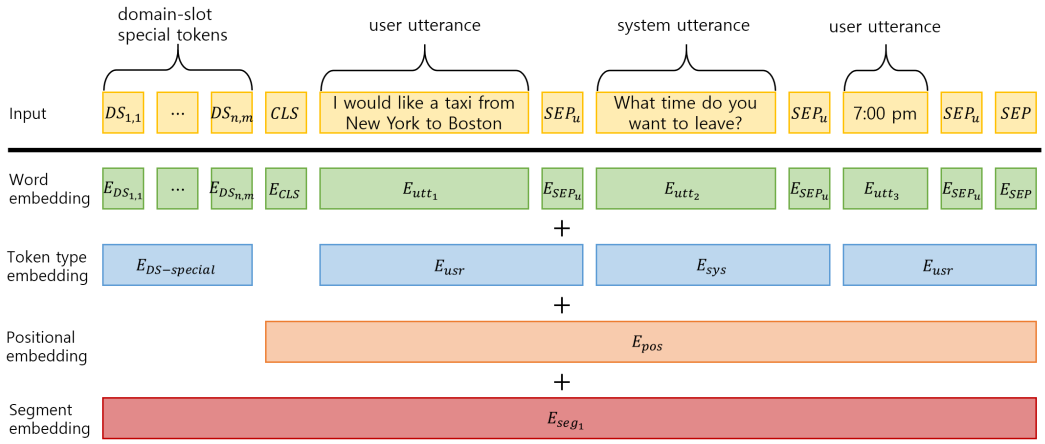

Figure 2: Input representation for the DSC encoder. The example here shows a dialog context for 2 turns of (S^1, U^1, S^2, U^2). S^1 is omitted because the sequence starts with the user utterance and S^1 is just a placeholder: a blank sentence. The special token layout is represented in *DS-merge* in this figure. E_{utt_n} represents the sequence of word embeddings for each utterance after tokenization.

Fig. 2 shows the input representation of the DSC encoder. The sequence begins with $\{DS\}$ tokens. The special token $[CLS]$ follows, which encodes the overall information of the dialog context. In the figure, we showed the *DS-merge* version. For the dialog context, we added a special token $[SEP_u]$ to separate each user or system utterance, which is added at the end of each utterance from the user or system. The input ends with a special token $[SEP]$ as the end-of-sequence token.

Notice that all $\{DS\}$ tokens run through the encoder together. In previous methods that encode domain-slot pair information with the dialog history, the dialog context is processed with individual domain-slot pairs in separate manner (Zhang et al., 2019; Lee et al., 2019; Wu et al., 2019). This method has its limitations can not model the relationship among domain-slot pairs because each of them are processed independently. This is especially a problem in multi-domain dialog, where different domain-slot pairs can share information. Also, this method costs much time and computation since every dialog history output has to be processed as many times as the number of domain-slot pairs. In our proposed method, all domain-slot pairs (or domains and slots) are run together. This enables modeling the relationship among different domain-slot pairs since different special tokens can interact with each other through the encoding process. Moreover, it is computationally more efficient than running the model for each domain-slot pair individually, as we previously mentioned the computational complexity of Transformer based models using our model ($O(n+k)^2$), and models that run each domain-slot pair separately with the dialog context ($O(k \times (n+1)^2)$).

4 types of embeddings are summed up to represent each token. We used the pre-trained word embedding of ALBERT, except for the domain-slot special tokens, which we randomly initialized. We introduced the token type embedding to differentiate the special domain-slot tokens, user utterances, and system utterances. For *DS-merge*, we used a single token type embedding to represent a domain-slot pair, whereas for *DS-split*, we used two token type embeddings for the domain and slot. Position embeddings are also employed from ALBERT, but the index of the positional embedding starts from the CLS token. We did not use the positional embedding for the domain-slot special tokens as the order within those tokens are meaningless. For the CLS token we did not apply this embedding. Lastly, the segment embedding from ALBERT was used to represent the sequence as a single segment.

DSC encoder encodes contextualized embeddings for every input token. However, for the slot-gate classifier and slot-value classifier, we only use the special token outputs of the DSC encoder ($[CLS]$ token and $\{DS\}$ tokens). This is formally defined as follows for *DS-merge* and *DS-split*, respectively, for turn t.

$$\widehat{D_1S_1}, \cdots, \widehat{D_nS_m}, \widehat{CLS} = DSCencoder([DS_{(1,1)}, \cdots, DS_{(n,m)}, CLS, X^t, SEP]), \quad (3)$$

$$\widehat{D_1}, \cdots, \widehat{D_n}, \widehat{S_1} \cdots, \widehat{S_m}, \widehat{CLS} = DSCencoder([D_1, \cdots, D_n, S_1 \cdots, S_m, CLS, X^t, SEP]), \tag{4}$$

where $DS_{(1,1)}, \cdots, DS_{(n,m)}$ indicates the domain-slot special tokens for *DS-merge* and $D_1, \cdots, D_n$ represent domain special tokens and $S_1, \cdots, S_m$ represent slot special tokens for *DS-split*. X^t represents the dialog context of $(S^1, U^1, SEP_u, \cdots, S^t, U^t, SEP_u)$, where U^t and S^t represents the utterance for the tth turn for the user and system respectively. S^1 is just a placeholder and the actually dialog starts with U^1. The $\{DS\}$ tokens and $[CLS]$ token with the hat notation() represents the encoded output of the DSC encoder for those special tokens. They are vectors of $\mathbb{R}^d$, where d is the hidden dimension of ALBERT. From now on, we assume a dialog context at time t from a dialog of T turns. For a single dialog, the same procedure is applied for $t = 1, \cdots, T$.

3.2 SLOT-GATE CLASSIFIER

For the slot-gate classifier, we use the DSC encoder output of the $\{DS\}$ tokens for each domain-slot pair to predict whether it is relevant to the dialog or not. In previous methods, gating used categories of $\{prediction, dontcare, none\}$, where $prediction$ means a slot value should be predicted for this domain-slot and $dontcare$ means that the predicted slot value is $dontcare$ and $none$ means that the domain-slot is non-relevant. However, the performance for the $dontcare$ category was far inferior over the other two categories, so we dismissed the $dontcare$ category and only used $\{prediction, none\}$. For the first special token layout version, the slot-gate classifier predicts the value using the domain-slot pair special token. For the domain-slot pair with the domain i and slot j, the slot-gate classifier output for *DS-merge* is

$$Gate_{D_iS_j} = sigmoid\big(W_{G_{DS_{(i,j)}}} \widehat{DS_{(i,j)}}\big), \tag{5}$$

where $W_{G_{D_iS_j}} \in \mathbb{R}^{1\times d}$. For *DS-split*, the slot-gate classifier uses the concatenated output of the corresponding domain and slot token. Similarly, for the same domain-slot pair of (i, j), the slot-gate classifier output for *DS-split* is

$$Gate_{D_iS_j} = sigmoid\big(W_{G_{(D_i,S_j)}} [\widehat{D_i}|\widehat{S_j}]\big), \tag{6}$$

where | represents concatenation of vectors and $W_{G_{(D_i,S_j)}} \in \mathbb{R}^{1\times 2d}$. The loss objective for the gate classification is as follows.

$$\mathcal{L}_{gate} = \sum_{(i,j)\in DS} BinaryCrossEntropy\big(y^{gate}_{DS_{(i,j)}}, Gate_{D_iS_j}\big), \tag{7}$$

where DS refers to the set of all domain-slot pairs and $y^{gate}_{D_iS_j}$ is the binary slot-gate label for domain i and slot j.

3.3 SLOT-VALUE CLASSIFIER

We employ the fixed-vocabulary based classification method for predicting slot values. As in (Zhang et al., 2019), the candidate-value list for each domain-slot pair was constructed by using the values from the training dataset, rather than using the incomplete ontology from the dataset. The CLS token is concatenated with the special domain-slot tokens, and used as the input to the slot-value classifier for each domain-slot pair. The slot-value classifier output with the domain i and slot j for *DS-merge* is as follows.

$$Slot_{D_iS_j} = softmax\big(W_{S_{DS_{(i,j)}}} [\widehat{DS_{(i,j)}}|\widehat{CLS}]\big), \tag{8}$$

where $W_{S_{DS_{(i,j)}}} \in \mathbb{R}^{n_{D_iS_j}} \times 2d$-way prediction layer for each domain-slot pair, where $n_{D_iS_j}$ is the number of candidate values for domain i and slot j. Similarly, for *DS-split*, the slot-value classifier output is

$$Slot_{D_iS_j} = softmax\big(W_{S_{(D_i,S_j)}}([\widehat{D_i}|\widehat{S_j}|\widehat{CLS}])\big), \tag{9}$$

where $W_{S_{(D_i,S_j)}} \in \mathbb{R}^{n_{D_iS_j}} \times 3d$.

The loss objective for the slot-value classification is as follows.

$$\mathcal{L}_{slot} = \sum_{(i,j)\in DS} CrossEntropy(y^{slot}_{D_iS_j}, Slot_{D_iS_j}), \tag{10}$$

where $y^{slot}_{D_iS_j}$ is the label for domain i and slot j.

3.4 Training objective and Model Summary

The DSC encoder, slot-gate classifier, and slot-value classifier are all jointly trained with the total loss objective.

$$\mathcal{L}_{total} = \mathcal{L}_{gate} + \mathcal{L}_{slot} \tag{11}$$

To summarize, for a given dialog context, the model first uses the slot-gate classifier to decide whether the dialog context is relevant to a certain domain-slot pair or not. Next, the slot-value classifier predicts the correct value among the candidate-values for all the domain-slot pairs that are relevant to the dialog context.

3.5 Training details

For the pre-trained language encoder encoder, we used the pretrained *large-v2* version of ALBERT(Lan et al., 2019) from (Wolf et al., 2019) in Pytorch (Paszke et al., 2019). We also used the tokenizer used for ALBERT, which is based on SentencePiece (Kudo & Richardson, 2018). We also included the model trained with a smaller size(*base-v2*) and larger size(*xlarge-v2*) version of ALBERT from the same source. The optimizer was AdamW (Loshchilov & Hutter, 2018) with a learning rate of $5.00\text{E-}05$. We also applied linear warm-up followed by linear decay for the learning rate. The original ALBERT was pre-trained with a sequence length of up to 512 tokens. Usually, the standard procedure for this situation is to truncate the sequence to 512 and discard the remaining tokens. However, to cover dialogs longer than 512 tokens that are in the dataset, we resized the positional embedding to cover a maximum length of the dialog. We preserved the original pre-trained position embedding for positions indices up to 512 and randomly initialized the remaining position indices. This method showed better results than limiting the maximum sequence length to 512.

4 Experiment setup and results

We evaluate our model using the joint goal accuracy, which considers a model prediction to be correct when the prediction jointly matches the ground truth values for all domain-slot pairs, given a dialog context.

4.1 Dataset

We use the MultiWOZ-2.1 dataset (Eric et al., 2019), which fixed noisy annotations and dialog utterances of the MultiWOZ 2.0 (Budzianowski et al., 2018). The dataset contains 7 domains and over 10,000 dialogs. We follow the previous studies and use 5 domains (train, restaurant, hotel, taxi, attraction) with 30 domain-slot pairs. The other two domains (police, hospital) have little data and do not appear in the test dataset. We follow the pre-processing explained in (Wu et al., 2019) and also used some minor pre-processing of merging words (e.g. include -s $\rightarrow$ includes) and adding apostrophes (it s $\rightarrow$ it's).

4.2 Results

Table 2 shows the joint goal accuracy of our model compred to previous methods. Among the models that we compare, TRADE (Wu et al., 2019) and DST-Picklkist (Zhang et al., 2019) does not use extra supervision other than the dialog context and dialog state, or domain-slot pair labels.

Other methods such as DSTQA (Zhou & Small, 2019) SST (Chen et al., 2020) uses additional extra supervision. SimpleTod (Hosseini-Asl et al., 2020) formulates a multi-task learning problem of solving dialog state tracking, action and response generation by using a single generative model. The action generation part involves additional supervision of the system action for domain-slots. The response generation part also involves a delexicalization process of the system response and access to the database and its search result.

Both of our models show better performance among models without any additional supervision other than the dialog context and domain-slot pair labels. Our model beats all other models using extra supervision except for one model and is in par in performance. The model with the *DS-merge* layout shows slightly better results than the model with the *DS-split* layout. The reason for this could be that slot sharing might not have been as effective as expected. Also, concatenation might not have been enough for merging the information form the domain and slot. We will investigate further into this matter in future studies. This is important in that scalability is much better for *DS-split* than *DS-merge*, as many slots can be shared across different domains, reducing the number of special tokens to be used.

Table 1: Results for the test dataset of the Multi-WOZ 2.1 Dataset. Extra supervision information is from (Hosseini-Asl et al., 2020)

Model	Extra Supervision	Joint Goal Accuracy
TRADE	-	45.6
DSTQA	knowledge graph	51.17
DST-Picklist	-	53.3
SST	schema graph	55.23
TripPy	action decision	55.3
SimpleTOD	system action, delexicalized system response	55.76
Proposed method *(DS-merged)*	-	**55.48**
Proposed method *(DS-split)*	-	**55.06**

Table 2: Results for the test dataset of the Multi-WOZ 2.1 Dataset. Extra supervision information is from (Hosseini-Asl et al., 2020)

Model	Extra Supervision	Joint Goal Accuracy
DSTQA	knowledge graph	51.17
SST	schema graph	55.23
TripPy	action decision	55.3
SimpleTOD	multi-task learning(+system action label)	55.76
TRADE	-	45.6
DST-Picklist	-	53.3
Proposed method *(DS-merged)*	-	**55.48**
Proposed method *(DS-split)*	-	**55.06**

4.3 Analysis

4.3.1 Size of the pre-trained language encoder

Table 3 shows the result for different versions of ALBERT. Clearly, a bigger language model shows better results as is shown in in downstream tasks that ALBERT was evaluated on (Lan et al., 2019).

4.3.2 Modeling relationship among different domain-slot pairs

To evaluate the effect of relationship modeling among different domain-slot pairs, we blocked the attention among different $\{DS\}$ tokens. We chose the ALBERT configuration of *large-v2* for the experiments. Table 4 shows that without the relationship modeling, our model performance deteriorates by a substantial amount. This validates our idea that relationship modeling is the crucial factor for our approach.

Table 3: Results for different ALBERT configurations

Model	ALBERT config	Joint Goal Accuracy
DS-merge	*base-v2*	55.01
	large-v2	55.48
	xlarge-v2	
DS-split	*base-v2*	53.54
	large-v2	55.06
	xlarge-v2	

Table 4: Results for ablation of domain-slot relationship modeling

Model	Joint Goal Accuracy
Proposed method (*DS-merged*)	**55.48**
w/o relationship modeling	**53.38**
Proposed method (*DS-split*)	**55.06**
w/o relationship modeling	**52.84**

5 Conclusion

In this paper, we propose a model for multi-domain dialog state tracking that effectively models the relationship among domain-slot pairs using a pre-trained language encoder. We introduced two methods to represent special tokens for each domain-slot pair, which is called *DS-split* and *DS-merge*. These special tokens work as a $[CLS]$ token specific to each domain-slot pair (*DS-split*) or domain and slot (*DS-split*), encoding information relevant to what it represents. Theses special tokens are run together with the dialog context through the pre-trained language encoder, which enables modeling the relationship among different domain-slot pairs. Our approach is computationally efficient since it does not run multiple times as the number of domain-slot pairs. The experimental results show that our model achieves state-of-the-art performance on the MultiWOZ-2.1 Dataset among models without any extra-supervision. Our model surpasses other models that use extra-supervision with an exception of one model that uses multi-task learning and additional system action labels. The ablation studies show that the relationship modeling among different domain-slot pairs is the key element of our model. Also, we showed that larger pre-trained language encoders show better performance. We hope to advance our research by finding ways to effectively apply our model towards the open-vocabulary approach, which will enable better generalization for candidate values that are outside of the training data.

Acknowledgments

Thanks to you all who are reviewing this submission preprint.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 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*, pp. 5016–5026, 2018.

Guan-Lin Chao and Ian Lane. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1907.03040*, 2019.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *AAAI*, pp. 7521–7528, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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), pp. 4171–4186, 2019.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*, 2019.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 264–273, 2019.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *Proc. Interspeech 2019*, pp. 1458–1462, 2019.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. Trippy: A triple copy strategy for value independent neural dialog state tracking. *arXiv preprint arXiv:2005.02877*, 2020.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*, 2020.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. Sumbt: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5478–5483, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Elnaz Nouri and Ehsan Hosseini-Asl. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Osman Ramadan, Paweł Budzianowski, and Milica Gašić. Large-scale multi-domain belief tracking with knowledge sharing. *arXiv preprint arXiv:1807.06517*, 2018.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 808–819, 2019.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*, 2019.

Victor Zhong, Caiming Xiong, and Richard Socher. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1458–1467, 2018.

Li Zhou and Kevin Small. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*, 2019.

A APPENDIX

You may include other additional sections here.