LION-Net: LIghtweight ONtology-independent Networks for Schema-Guided Dialogue State Generation

Kai-Ling Lo Ting-Wei Lu Tzu-teng Weng I-Hsuan Chen Yun-Nung Chen

No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan
ianlo0511@gmail.com, {b05602022, r07922118, b05505046}@ntu.edu.tw, y.v.chen@ieee.org

Abstract

Domain-independent dialogue state tracking has received much attention in the recent studies of task-oriented dialogue systems. In this paper, we propose a LIghtweight ONtology-independent (LION) sequence-to-sequence model with copy and attention mechanisms to tackle the zero-shot dialogue state tracking problem. By sharing the parameters across domains and using the natural language descriptions of the services, intents, and slots as the input, our model enables zero-shot generalization to an unseen domain. The experiments are conducted on the newly-released schema-guided dialogue dataset (Rastogi et al. 2019), and our model outperforms the Google baseline on most metrics while requiring considerably less computational cost, both in memory usage and training time.

Introduction

The task-oriented dialogue system is the core technology of virtual assistants such as Amazon Alexa, Apple Siri, and Google Assistant. It can reduce the operating cost by automating online customer service.

Dialogue state tracking (DST), maintaining user's intentional states throughout the dialogue, plays a crucial role in task-oriented dialogue systems such as flight booking or restaurant reservation (Young et al. 2013). The goal of DST is to extract user goals or intentions mentioned across the conversations and to record them as a set of dialogue states, which is often modeled as a set of slots and their corresponding values. The dialogue system then can plan the next action from the record and the current state and respond to the user. For example, with a given domain "movie" and a slot type "genre", the system predicts the probability of corresponding slot-value candidates (e.g., "*horror*", "*comedy*").

In traditional approaches, state tracking utilized predefined ontology where all slots and their values are known, simplifying DST into a multi-class classification problem (Henderson, Thomson, and Williams 2014; Mrkšić et al. 2017; Zhong, Xiong, and Socher 2018). This approach, however, is incapable of dealing with a dynamic set of slots and intents. It is difficult to adapt the model to new domains.

The eighth Dialog System Technology Challenges (DSTC8) proposed a schema-guided paradigm as well as a schema-guided dialogue dataset (Seokhwan Kim 2019). The main goal is to seek possible zero-shot state tracking methods by making predictions with the natural language descriptions of a dynamic set of intents and slots. Specifically, given a schema, we predict the slot values by utilizing the natural language descriptions of both the schema and slots. As a result, we can generalize the model to an unseen service (schema) provided that we have the natural language description of the service and slot.

Motivated by TRADE (Wu et al. 2019), we propose a lightweight sequence-to-sequence model with both the attention mechanism and copy mechanism for the schema-guided state tracking task. Our model is shown to be effective for the schema-guided dialogue dataset while being reasonably efficient, requiring little memory usage and little training time.

Task Description

The task is to generate a dialogue state based on conversational contexts and the schema provided by specific service. Conversational contexts are composed of system utterances and user utterances. The state of dialogue, only present in user turns, consists of active intent, requested slots, and slot values.

Active intent is the intent in the current turn or what the user determines to do now. Requested slots are a list of slots requested by the user in the current turn. Slot values are values of slots until the current turn.

Schema of a specific service is made up of a set of intents and slots along with their natural language descriptions. The dialogue state needs to be predicted over these intents and slots, based on dialogue contexts.

Dataset

The dataset we used is the schema-guided dialogue (SGD) dataset collected by Google (Rastogi et al. 2019), which is composed of conversations between a virtual assistant and a user. These conversations have been generated with the help of a dialogue simulator and paid crowd-workers. The dataset contains a far larger number of domains, slots,

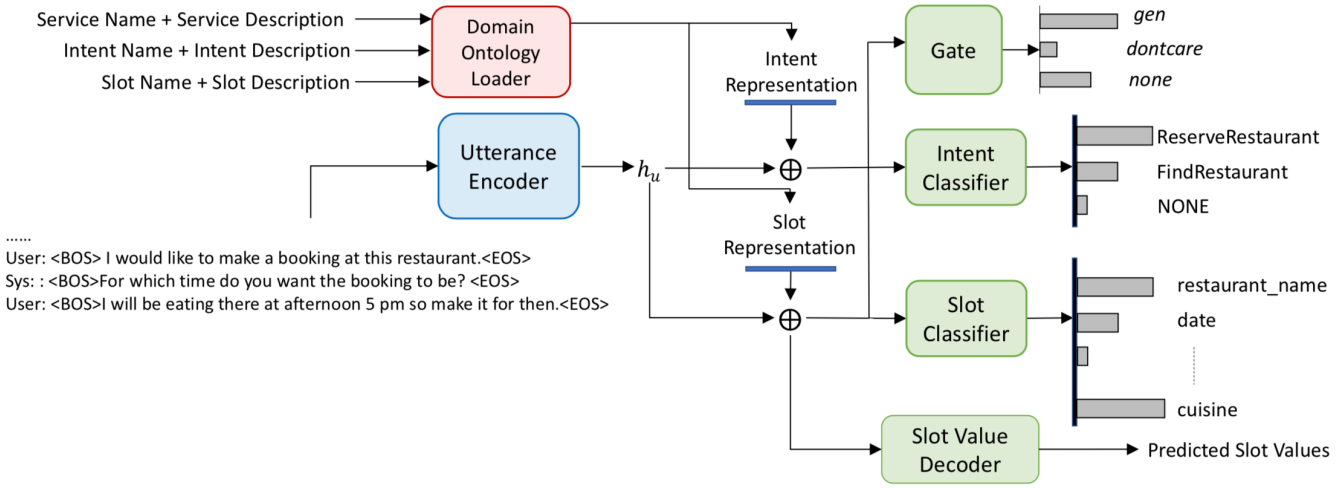

Figure 1: The illustration of the proposed model framework, which contains an utterance encoder for encoding dialogues, a decoder for predicting the slot values, two classifiers, one for prediction of active intents and another for prediction of requested slots, a domain ontology loader for generating the representations of slot and intent for the corresponding service, and a gate module to decide whether we should take the value from prediction or not. $\oplus$ denotes vector concatenation.

and slot values compared to other datasets like MultiWOZ (Budzianowski et al. 2018), a multi-domain and task-oriented dialogue dataset. Its evaluation sets contain many services and slots which are not present in the training set, to help evaluate model performance on unseen services. With more than 16000 dialogues in the training set, it is considered the largest publicly available labeled task-oriented dialogue dataset. The labels include the active intents and dialogue states for each user utterance and the system actions for every system utterance. It has both single-domain and multi-domain dialogues.

Related Work

Traditional dialogue state tracking models estimate the present dialogue states given a previous semantics extracted by language understanding modules (Williams and Young 2007; Thomson and Young 2010; Wang and Lemon 2013; Williams 2014), or directly infer the state by the conversation history and the user utterance (Henderson, Thomson, and Williams 2014; Zilka and Jurcícek 2015; Mrkšić et al. 2015). However, these models are difficult to extend to new domains because they rely on handcrafted semantic dictionaries and delexicalization to reach generalization. To address the above problems, Mrkšić et al. (2017) proposed distributional representation learning to leverage semantic information from word embeddings as opposed to hand-crafting features but the scalability still remains as a challenge. When estimating the state of the dialogue, Mrkšić et al. (2017) individually score all slot-value combinations. It is not practical in the real world services having a very large and dynamic set of possible slot values.

In recent work, SUMBT (Lee, Lee, and Kim 2019) encoded utterances, domain-slot-type and slot-value literally with BERT (Devlin et al. 2018). It uses an attention mechanism to retrieve relevant information from the utterances corresponding to domain-slot-type. The non-parametric way to predict the slot values enables the model not to structurally depend on domains and slot-types. Therefore, the model can deal with unseen services and utilize shared knowledge among multiple domains and slots.

TRADE (Wu et al. 2019) used a sequence to sequence recurrent neural networks with a copy mechanism to generate the slot value for a given slot name. The copying mechanism and context-enhanced gate help the model to overcome the multi-turn mapping problem. Besides, the model can share knowledge between domains to trace unseen slot values by sharing its parameters. It can also adapt to new few-domains since the model leverages the domains it has already seen during training.

Both models are scalable and able to predict the dynamic set of slot values. They achieve good performance on MultiWOZ (Budzianowski et al. 2018) dataset and the latter is the current state-of-the-art model. However, it is difficult to directly apply the above-mentioned models on SGD dataset. Since the SGD dataset contained both categorical and non-categorical slot values, MultiWOZ has only categorical slot values. Moreover, the SGD dataset has a much larger scale and the presence of multiple services per domain. In this paper, the model we proposed is based on the original TRADE (Wu et al. 2019) model. The results are shown in the evaluation section.

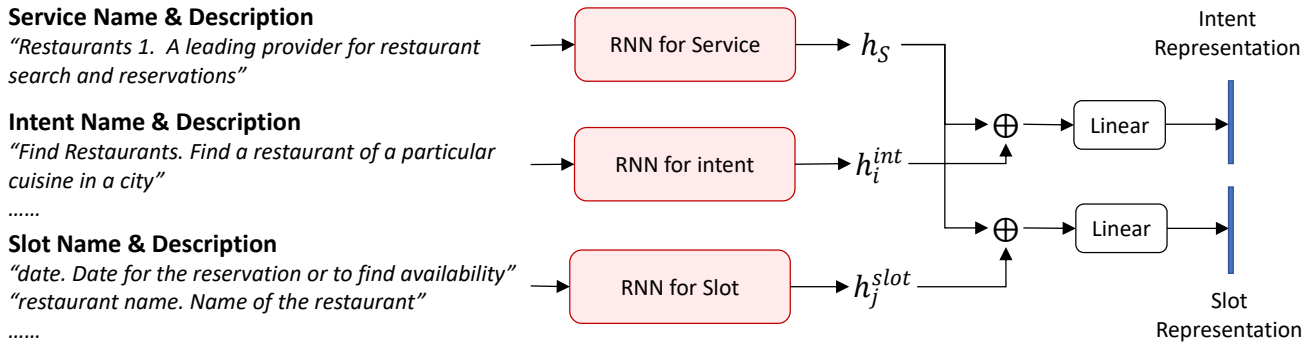

Figure 2: Model architecture for the domain ontology loader, which generates the representations of service, intents, and slots. By default, we use 3 different split RNNs to encode service, intents, and slots. We concatenated name and description as input to RNN for all of the services, intents, and slots. Experiments were also done using only one shared RNN for the representation of service, and corresponding intents and slots. RNN for service can be disabled, so that the vector for slot(or intent) is only made up of intent and slot natural language name and description, containing no service information. $\oplus$ denotes vector concatenation.

Proposed Approach

For each conversation turn, the model takes dialogue history $\mathbf{H}$ and service schema $\mathbf{S}$ as the input and outputs the dialogue state $\mathbf{V}$. Specifically, $\mathbf{S}$ is composed of a service S, a set of intents $\{\text{Int}_i\}_1^{T_i}$, and a set of slots $\{\text{Slot}_j\}_1^{T_s}$. T_i is the number of intents in service S and T_s is the number of slots in service S.

Our model comprises several components: an utterance encoder, a schema encoder (or domain ontology loader), a slot gate, two classifiers for active intent and requested slots, and a state generator(or slot value decoder), which generate the final value for each possible slot illustrated in Figure 1. Our model can be seen as an extension of TRADE (Wu et al. 2019), and each component is detailed below.

Utterance Encoding The utterance encoder, which is an RNN, encodes the dialogue history into a sequence of fixed-length vectors. For a given dialogue history $\mathbf{H}$ composed of utterances $\{u_1, u_2, ..., u_L\}$, where L is the length of the dialogue, we flatten the whole history into one sequence, which is $\{w_1, w_2, ..., w_{T_u}\} = \{w_{1,1}, w_{1,2}, ..., w_{L,l_L}\}$, in which $u_i = \{w_{i,1}, ..., w_{i,l_i}\}$, l_i is the length of u_i. We feed the sequence into utterance encoder to get output vectors $O = \{o_1, o_2, ..., o_{T_u}\}$ and final hidden state h_u. T_u is the total length of the sequence.

Schema Embedding We encode the schema information using the description given for every service, intent and slot. The schema encoder comprises of three RNNs, or only one RNN shared by three types of description. The RNN encodes the given description $\{w_1, w_2, ..., w_d\}$ into a fixed-length representation h, where d is the length of description. We annotate the representation of service, intents and slots as h_s, $\{h_i^{\text{int}}\}_1^{T_i}$, and $\{h_j^{\text{slot}}\}_1^{T_s}$.

Active Intent We predict the active intent using embeddings from schema encoder and the hidden state from utterance encoder. We concatenate h_u, h_s and h_i^{int} for given service and each intent, then feed them into a linear layer L_I

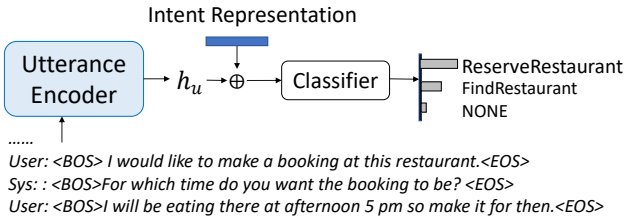

Figure 3: Illustration of the active intent classifier, where the classifier is composed of linear layer(s) and a softmax layer.

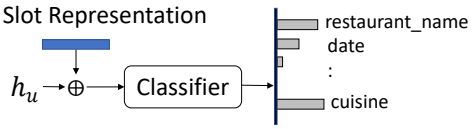

Figure 4: Model architecture for the requested slot classifier. The classifier is composed of linear layers and a sigmoid layer. h_u is the last hidden state of utterances encoder. $\oplus$ denotes vector concatenation.

to get the score of each intent. We pick the intent with highest score as the active intent for current turn. The distribution is defined as

$$P^I = \text{Softmax}(\left[L_I([h_u, h_s, h_i^{\text{int}}])\right]_1^{T_i}) \in \mathbb{R}^{T_i}$$

Requested Slots For requested slots, we utilize the schema embeddings and utterance hidden state in the same manner as we predict active intent. After getting the score of each slot, we pass them through sigmoid function and pick the slots of which the output larger than 0.5 as the requested slots for the current turn. The distribution is defined as

$$P^R = \text{Sigmoid}(\left[L_R([h_u, h_s, h_j^{\text{slot}}])\right]_1^{T_s}) \in \mathbb{R}^{T_s}.$$

Slot Prediction As in (Wu et al. 2019), we use a state generator and a context-enhanced slot gate to generate value for

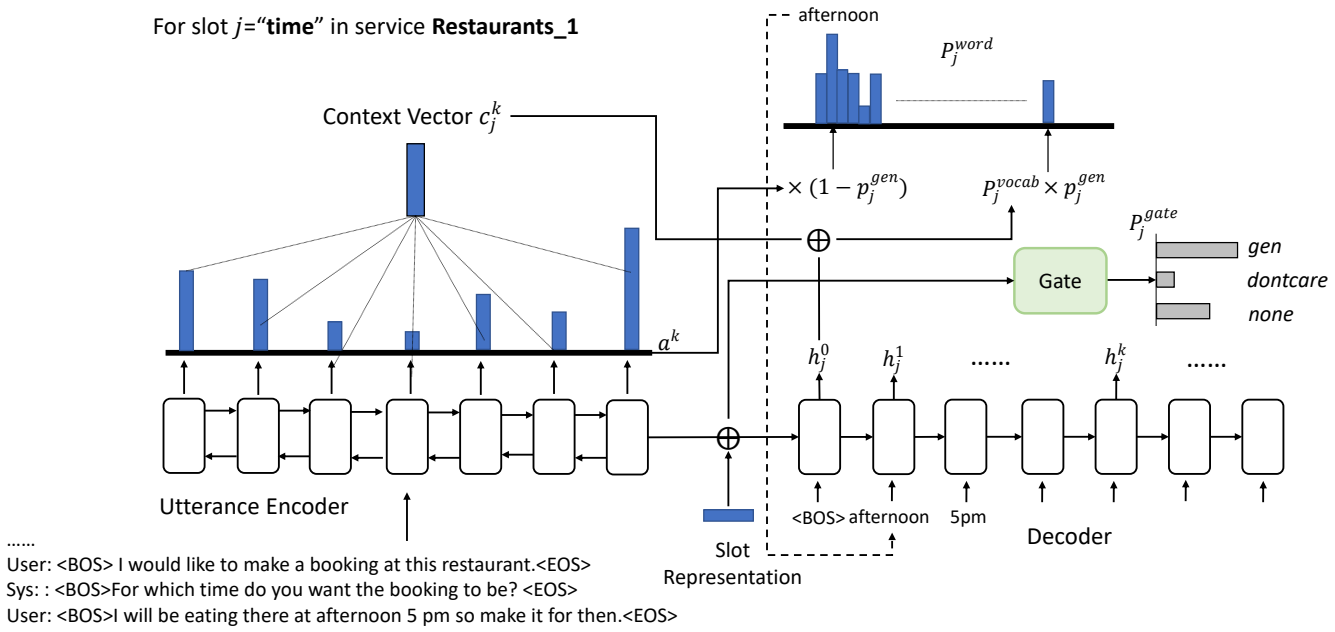

Figure 5: Model architecture for slot filling. We encode the dialogue history through a bidirectional RNN to generate the representation of the whole dialogue until the current turn. Then we concatenate the last hidden state of the encoded dialogue history and the vector for slot produced in Figure 1 to be the initial hidden state of the decoder (or state generator). $\oplus$ denotes vector concatenation.

each possible slot. The state generator decodes output tokens for all possible *(service, slot)* pairs; the context-enhanced slot gate predicts whether each of the slots has an actual value via a three-way classifier illustrated in Figure 5.

Given the history utterances H, service S and slot Slot_j, we pass the concatenation of vectors h_u, h_s and h_j^{slot} to the linear gate to predict the class of value of the slot Slot_j from three classes, *gen*, *none* or *dontcare*.

$$P_j^{\text{gate}} = L_g([h_u, h_s, h_j^{\text{slot}}]) \in \mathbb{R}^3,$$

where $j = 1, 2, 3, ...,$ number of slots corresponding to service S. If the slot gate predicts *none* or *dontcare*, then we ignore the value decoded by state generator and fill the slot as empty or "don't care". Otherwise, we take the value from generator as our final prediction.

Following TRADE (Wu et al. 2019), we use a GRU as the main component of our state generator(or the decoder), and the decoder will generate values for T_s pairs independently. Unlike original design, we pass the schema information through projection L_p of concatenated vector $[h_s, h_j^{\text{slot}}]$ to same dimension as hidden state h_u from utterance encoder and add the projected vector to h_u as the initial state. That is, the initial hidden state of generator is

$$h_0 = h_u \oplus L_p([h_s, h_j^{\text{slot}}])$$

in which $\oplus$ indicates element-wise addition.

At each decoding step k , the generator takes a word embedding w_j^k as input, and outputs a hidden state h_j^k. An attention distribution a^k is calculated based on hidden state and outputs from utterance encoder:

$$e_i^k = v^T \tanh(W_o o_i + W_h h_j^k + b_{\text{attn}})$$
$$a^k = \text{Softmax}(e^k)$$

where v, W_o, W_h, b_{attn} are trainable parameters. The distribution a^k is used to generate a context vector c_j^k for each decoding step,

$$c_j^k = \sum_i a_i^k o_i$$

and also can be seen as a distribution over the flattened utterance history. We pass the context vector and generator hidden state through two linear layers to produce a vocabulary distribution P_j^{vocab}:

$$P_j^{\text{vocab}} = \text{Softmax}\Big(L_{v,2}\big(L_{v,1}([c_j^k, h_k])\big)\Big)$$

In addition, a generation probability p_{gen} is calculated from the decoded input w_j^k, generator hidden state h_j^k and context vector c_j^k:

$$p_j^{\text{gen}} = \text{Sigmoid}(w_c^T c_j^k + w_d^T h_j^k + w_x^T w_j^k + b_{\text{ptr}})$$

where w_c, w_d, w_x and b_{ptr} are trainable parameters. The decoded input w_j^k is decoded from previous generator hidden state h_j^{k-1}. The final output distribution is calculated as weighted summation of P_{vocab} and a^k:

$$P_j^{\text{word}} = p_j^{\text{gen}} * P_j^{\text{vocab}} + (1 - p_j^{\text{gen}}) * a^k.$$

Training During training, we have four sources of loss: the distribution of active intent P_I, the distribution of requested slots P_R, the output of slot gate P_{gate} and the word distribution P_{word} . We optimize the model with these four losses simultaneously. First, a cross-entropy loss L_I is computed between P^I and one-hot label y^I,

$$L_I = -\log(P^I \cdot (y^I)^T).$$

Second, a binary-entropy loss L_R is computed between P^R and multi-hot label y^R,

$$L_R = -\log(P^R \cdot (y^R)^T).$$

For the slot gate and state generator, we have cross-entropy loss L_{gate} and L_{word} as below:

$$L_{\text{gate}} = \sum_{i=1}^{T_s} -log(P_i^{\text{gate}} \cdot (y_i^{\text{gate}})^T)$$

$$L_{\text{word}} = \sum_{i=1}^{T_s} \sum_{j=1}^{|y_i^{\text{word}}|} -\log(P_{i,j}^{word} \cdot (y_{i,j}^{\text{word}})^T).$$

We optimize the weighted sum of these four losses using hyper-parameters α, β, γ and δ ,

$$L = \alpha L_I + \beta L_R + \gamma L_{\text{gate}} + \delta L_{\text{word}}.$$

Configurations We set the vocabulary size to be 5000 and the embeddings are initialized with 300-dimension GloVe(Pennington, Socher, and Manning 2014). The vocabulary contains the most frequently present 5000 words in both training and dev set. The model is trained using Adam optimizer(Kingma and Ba 2014) with a batch size of 32. We perform several experiments with different hyper-parameters. The best performing setting is with a learning rate of 0.001, a dropout rate of 0.1. For loss function, 0.4 is set for alpha and beta, 0.1 is set for gamma and delta.

Evaluation

We use the following metrics for evaluation of the dialogue state tracking task:

- **Active Intent Accuracy:** The fraction of user turns where the active intent has been correctly predicted.
- **Requested Slot F1:** The macro-averaged F1 score for requested slots overall eligible turns. Turns with no requested slots in ground truth and predictions are skipped.
- **Average Goal Accuracy:** For each turn, we predict a single value for each slot present in the dialogue state. The slots which have a non-empty assignment in the ground truth dialogue state are considered for accuracy. This is the average accuracy of predicting the value of a slot correctly. A fuzzy matching score is used for non-categorical slots to reward partial matches with the ground truth.
- **Joint Goal Accuracy:** This is the average accuracy of predicting all slot assignments for a turn correctly. For non-categorical slots, a fuzzy matching score is used.

All of the teams joining the competition used the same evaluation metrics. Google has provided an evaluation script for the consistency of evaluation results.

Figure 6: Best performance of the model on the validation set on all services, services seen in training data, services not seen in training data. $\oplus$ denotes vector concatenation.

Performance on SGD dataset The model performs well for Active Intent Accuracy and Requested Slots F1 across both seen and unseen services. For joint goal and average goal accuracy, the model has better performance on seen services compared to unseen services (Figure 6). We think one reason for this is that the OOV rate is higher for slot values in unseen services.

The results are similar to the Google baseline. (Rastogi et al. 2019) However, our model performs better on seen services and performs worse on unseen services compared to the Google baseline model.

An example of state predicting in unseen services(here the service is **Alarm_1**, which is not seen in training set) is shown in Figure 7. The active intent was predicted wrong as **AddAlarm** when the reference active intent is **GetAlarm**. Furthermore, our model tends to predict more slots in the predicted state than in the reference state. A possible explanation is that the slot descriptions of slot names defined in the schema are ambiguous so that our model cannot distinguish between them. For example, the description of **alarm_time** is "Time of the alarm" and the description of **new_alarm_time** is "Time to set for the new alarm"; the difference between them is not clear even for human.

Performance on different domains The performance on the validation set on different domains is shown in Table 3. We can observe that one of the factors affecting the performance across domains is still the presence of the service in the training data(seen services), similar to Google baseline (Rastogi et al. 2019). Performance on seen services is generally better than performance on unseen services. Among seen services, domain 'Homes' has the best performance on both Joint Goal Accuracy and Average Goal Accuracy. Our model performs better than Google baseline in most of the seen services.

For unseen services, we observe a large difference between Joint Goal Accuracy and Average Goal Accuracy. The reason might be that we did not differentiate categorical slots and non-categorical slots during training and prediction. There are a large number of non-categorical slots in

Model	Active Int Acc	Req Slot F1	Avg Goal Acc	Joint Goal Acc	Training time per epoch	GPU Memory Usage	Number of trainable Params
TRADE (Wu et al. 2019)	-	-	-	0.23	5 days	20G	10.22M
Google Baseline (Rastogi et al. 2019)	0.885	0.972	0.694	0.383	12 hrs	8G	118.35M
Our Proposed Model	0.974	0.975	0.851	0.57	2 hrs	1.8G	6.29M

Table 1: Experimental results on the validation set on all services with different models. GA means goal accuracy. We also compare the training time for 1 epoch, GPU memory usage for training and number of trainable parameters. The GPU we used for our proposed model and Google baseline is GeForce RTX 2070 and for TRADE we used Tesla P40. The number of trainable parameters are also compared, and our model has significantly less number of trainable parameters than the other two.

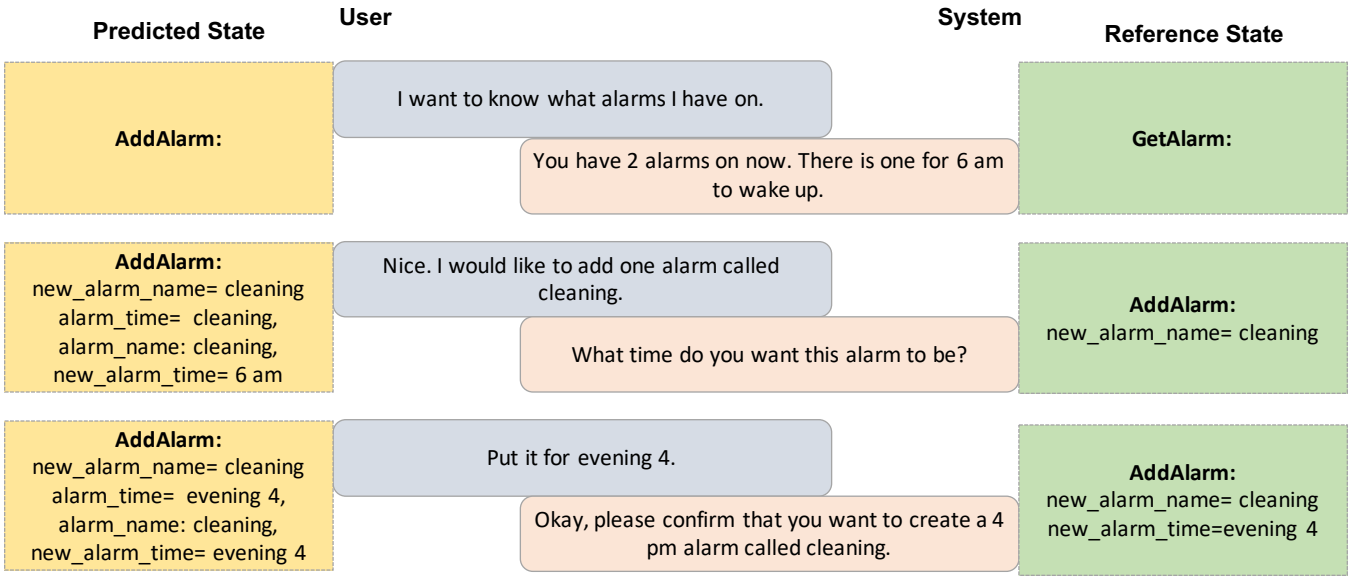

Figure 7: Comparison of predicted dialogue states and reference dialogue states in a dialogue in validation set.

Experiment Name	Active Int Acc	Req Slot F1	Avg GA	Joint GA
RNN for service disabled, split RNNs	0.924	0.976	0.744	0.365
RNN for service enabled, split RNNs	0.935	0.970	0.738	0.354
RNN for service disabled, shared RNNs	0.917	0.975	0.715	0.312
RNN for service enabled, shared RNNs	0.907	0.965	0.697	0.319

Table 2: Experimental result on test set on all services. GA means goal accuracy.

domains in unseen services(such as 'amount' in 'Banks', or 'date' in 'Restaurants'.) It's difficult for the decoder to decode the correct "amount" or "date" without using the span of utterance.

Comparing different models We have used original TRADE (Wu et al. 2019) as the baseline but we found that training time is too long(train 1 epoch for 5 days) so we only reported joint goal accuracy trained for 1 epoch, which is 0.23 on the validation set on all services(table 1). One reason for poor performance might be inadequate training time, and another might be that the original TRADE was designed for the MultiWOZ dataset not for the SGD dataset. The original TRADE model predicts slot values for every slot in every service in each turn, making it too slow to train and consuming a lot of memory. It does not use the annotated 'service' in the state of the user as a guide to reduce the iteration of every possible slot. Our model is much more efficient(training 1 epoch for 2 hrs.) since we have used the annotated 'service' in the state of the user as a guide to reduce the prediction space of the slot values.

Our model performs better over the Google baseline on the validation set. A possible key difference is that we encode the whole dialogue history but Google baseline only used the last two utterances as utterance encoding. Our utterance encoding contains more context information than that of the Google baseline. The best performance of our model still exceeds the Google baseline on the test set for all services. However, we perform worse than Google baseline only in joint goal accuracy (our model 0.365 vs Google baseline 0.411), and the probable reason might be that there are more unseen services in the test set. The Google base-

Domain	Joint GA	Avg GA
Alarm*	0.175	0.651
Banks*	0.103	0.875
Flights*	0.317	0.825
Hotels**	0.707	0.947
Media*	0.229	0.646
Movies*	0.330	0.583
Restaurants*	0.103	0.710
Services*	0.165	0.646
Buses	0.899	0.973
Events	0.822	0.966
Homes	0.900	0.979
Music	0.622	0.879
RentalCars	0.767	0.948
RideSharing	0.766	0.915
Travel	0.808	0.944
Weather	0.862	0.95

Table 3: Best model performance on the validation set per domain. (GA: goal accuracy). Domains marked with '*' are those for which the service in the dev set is not present in the training set. Hotels domain marked with '**' has one unseen and one seen service. For other domains, the service in the dev set was also seen in the training set.

line model uses contextualized word embedding, which has been shown to significantly outperform traditional word embedding on downstream tasks, as their encoder, at the cost of larger memory usage and longer training time. Our model, on the other hand, uses word embedding for its training, thus taking less memory, less training time, and less inference time, which is crucial since the response time of the virtual assistant is a key for the real-world usage.

Table 3 shows our performance of each domain on the validation set, where the marked domains are unseen. We observe that our model performs better for domains containing seen services in the training data. The result is similar to Google baseline model (Rastogi et al. 2019).

Conclusion

In this paper, we have proposed a scalable, relatively lightweight model to track the state of the dialogue. The model outperforms the Google baseline on the validation set on all evaluation metrics. The model uses less training time and less memory compared to the original TRADE and Google baseline model but achieves better or similar results.

References

Budzianowski, P.; Wen, T.-H.; Tseng, B.-H.; Casanueva, I.; Ultes, S.; Ramadan, O.; 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*, 5016–5026. Brussels, Belgium: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 263–272. Philadelphia, PA, U.S.A.: Association for Computational Linguistics.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, H.; Lee, J.; and Kim, T.-Y. 2019. Sumbt: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 5478–5483.

Mrkšić, N.; Ó Séaghdha, D.; Thomson, B.; Gašić, M.; Su, P.-H.; Vandyke, D.; Wen, T.-H.; and Young, S. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 794–799. Beijing, China: Association for Computational Linguistics.

Mrkšić, N.; Ó Séaghdha, D.; Wen, T.-H.; Thomson, B.; and Young, S. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1777–1788. Vancouver, Canada: Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Rastogi, A.; Zang, X.; Sunkara, S.; Gupta, R.; and Khaitan, P. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Seokhwan Kim, Michel Galley, C. G. S. L. A. A. B. P. H. S. J. G. J. L. M. A. M. H. L. L. J. K. K. W. S. L. C. H. A. C. T. K. M. A. R. X. Z. S. S. R. G. 2019. The eighth dialog system technology challenge. *arXiv preprint*.

Thomson, B., and Young, S. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Comput. Speech Lang.* 24(4):562–588.

Wang, Z., and Lemon, O. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL Conference*.

Williams, J. D., and Young, S. 2007. Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.* 21(2):393–422.

Williams, J. D. 2014. Web-style ranking and SLU combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 282–291. Philadelphia, PA, U.S.A.: Association for Computational Linguistics.

Wu, C.-S.; Madotto, A.; Hosseini-Asl, E.; Xiong, C.; Socher, R.; and Fung, P. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *The 57th Annual*

Meeting of the Association for Computational Linguistics (ACL 2019).

Young, S.; Gašić, M.; Thomson, B.; and Williams, J. D. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.

Zhong, V.; Xiong, C.; and Socher, R. 2018. 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)*, 1458–1467. Melbourne, Australia: Association for Computational Linguistics.

Zilka, L., and Jurcícek, F. 2015. Incremental lstm-based dialog state tracker. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* 757–762.