Memory Attention Neural Network for Multi-Domain Dialogue State Tracking

Zihan Xu[1,2] *, Zhi Chen[1,2] *, Lu Chen[1,2†], Su Zhu[1,2], and Kai Yu[1,2†]

[1] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[2] SpeechLab, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
{zihan.xu, zhenchi713, chenlusz, paul2204, kai.yu}@sjtu.edu.cn

Abstract. In a task-oriented dialogue system, the dialogue state tracker aims to generate a structured summary (*domain-slot-value* triples) over the whole dialogue utterance. However, existing approaches generally fail to make good use of pre-defined ontologies. In this paper, we propose a novel Memory Attention State Tracker that considers ontologies as prior knowledge and utilizes Memory Network to store such information. Our model is composed of an utterance encoder, an attention-based query generator, a slot gate classifier, and ontology Memory Networks for every domain-slot pair. To make a fair comparison with previous approaches, we also conduct experiments with RNN instead of pre-trained BERT as the encoder. Empirical results show that our model achieves a compatible joint accuracy on MultiWoz 2.0 dataset and MultiWoz 2.1 dataset.

1 Introduction

Dialogue State Tracking (DST) is a key part of task-oriented dialogue systems, which has attracted more and more attention [2]. DST is expected to accurately summarize the intentions of a user and extract a compact representation of dialogue states. The dialogue state is a set of *domain-slot-value* triples, which records all the user's conditions (e.g., *train-leaveat-13:45* means that the user wants to order the train's ticket leaving at 13:45.).

Traditional approaches for DST rely on hand-crafted rules and rich expert knowledge [18]. Recently, data-driven dialogue state trackers achieve excellent performance improvement. Current deep-learning based models can be taken into two categories: classification and generation. Classification approaches [21, 20] assume that all ontologies are predefined in the task, where dialogue state trackers only need to consider DST as a classification problem. For example, for slot *bookday* in domain *train*, which tells us when the train leaves, all possible values can be known in advance. Generation approaches use an open-vocabulary setup for value searching [19]. They assume that some values can be copied from dialogue context and all values can be extracted from an open-vocabulary.

* Zihan Xu and Zhi Chen are co-first authors and contribute equally to this work.
† The corresponding authors are Kai Yu and Lu Chen.

However, it is difficult for generation approaches to handle the problem when complex reasoning is required to get the right answer.

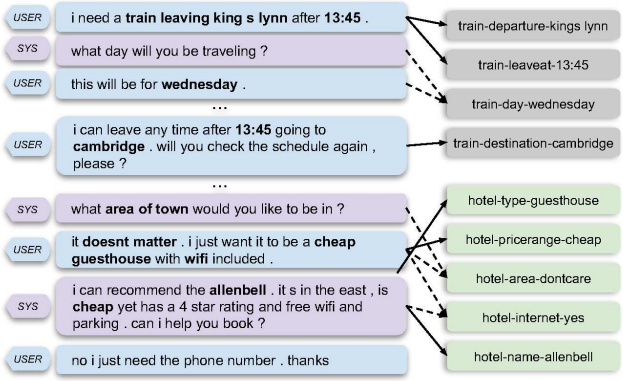

Fig. 1. An example of multi-domain dialogue state tracking

A large scale multi-domain dialogue dataset (MultiWOZ 2.0) was recently introduced by [14], and MultiWoz 2.1 [7] was proposed by fixing many annotation errors in MultiWoz 2.0. These datasets are two of the largest multi-domain dialogue datasets available at present. There are mainly two challenges that make the task difficult. The first challenge is the cross-domain character. As shown in Figure 1, a user can first ask the system about booking a train and then transfer to asking about hotel rooms, which requires flexibility in the ability of domain-transition. Moreover, slots like *train-departure* and *train-destination* are quite confusable since they share a lot of similarities. In total, there are 1879 possible values in MultiWoz 2.0 and 1934 in MultiWoz 2.1, which creates a large number of combinations. In this case, our model has to first determine the correct domain-slot pairs to track, then accurately point out the right values. Another challenge is that complex reasoning is required in multi-turn dialogues. As in Figure 1, the dot arrows show how multi-turn reasoning may be needed throughout a dialogue. For example, state *hotel-area-dontcare* contained in this dialogue, we even need to make cross-domain reasoning to get the right prediction.

In this work, we formulate the dialogue state tracking as a classification problem based on predefined ontologies. We employ the Memory Network to store the predefined ontologies and use an attention-based query generator to generate a slot-aware summary of the dialogue context. Our model resolves dialogue state tracking tasks into three separate sub-tasks: (1) dialogue summary generation, which generates a query based on attention scores between domain-slot pair and dialogue utterance, (2) slot value prediction, which calculates a probability distribution over possible values stored in memory layers and updates the query for each hop, and (3) slot state prediction, which predicts what the slot

state is among {*value, don't care, none*}. By sequentially updating the query, both dialogue information and ontology knowledge are integrated. This is quite intelligible because sometimes we need to traverse all the possible values in (2) to determine the answer in (3).

In order to improve the reasoning ability of dialogue state tracker, we propose a novel architecture by incorporating predefined ontologies into memory networks. Memory Network [17] is a general paradigm used in machine reading tasks. In this work, we employ the Gated Memory Network [13], which adds Memory Gates between adjacent memory layers.

We propose the Memory Attention Neural Network for multi-domain task-oriented dialogue tracking. Explainability and compatibility are the two main advantages of our proposed model. Contributions in this work are summarized as:

- We utilize prior knowledge by setting up a state classifier Memory Network, which is the first work to model the predefined ontologies in Dialogue State Tracking.
- We build a computational efficient multi-domain DST by proposing an architecture that is implemented purely with the Attention Mechanism, which also shows a performance boost.
- We adopt recent progress in large pre-trained contextual word embeddings [6] into dialog state tracking, and get compatible performance.
- We conduct ablation studies on memory hops and memory gates and provide a comprehensive error analysis of the results.

2 Related Work

Dialogue State Tracking Traditional methods of dialogue state tracking are often based on hand-crafted rules [8], where the principal focus is on building a set of semantic parsing rules. These rule-based methods have been widely used in commercial systems. However, the performance of traditional models relies heavily on a huge amount of expert knowledge, which is labor-intensive and expensive. Besides, due to the natural limitation of hand-crafted rules, traditional methods are inclined to make mistakes in real practice.

Statistical dialog state trackers calculate a probability distribution over hypotheses of the real dialogue state, based on the outputs of semantic parsing module. Compared with traditional methods, statistical models are more robust in many scenarios. Recent approaches use end-to-end architectures[3, 9, 19] to combine Natural Language Understanding and Dialogue State Tracker. These approaches can be further classified into fixed-vocabulary and open-vocabulary. Fixed-vocabulary approaches usually assume that all ontologies are known in advance and the model only has to pick the correct value from ontologies. Open-vocabulary approaches often start with zero knowledge of possible values and generate the candidate values from an open-vocabulary.

Memory Networks based DST Memory Network has been proven useful in machine reading tasks, where the model is capable to perform complex reasoning. Some previous works have used Memory Networks in a dialogue state tracking task. For example, in an early work [16], dialogue state tracking was formulated as a machine reading task, where retrieving dialogue state is completed by a reading comprehension task.

In this work, we use Memory Network as the decoder for dialogue state tracker. We use a dialogue encoder to generate a query for the input of the Memory Network. We use the attention scores derived at the last hop as a probability distribution of dialogue states, and we use the query from the last hop for dialogue state gate classification.

3 Proposed Framework

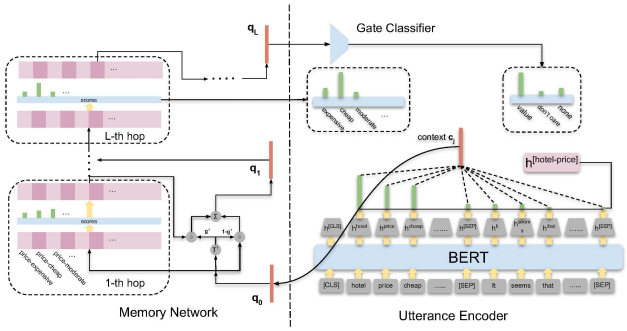

Fig. 2. An overview of our proposed model

In a multi-domain dialogue system, there are a set of domains $\mathcal{D}$ that users and the system can converse about. For each domain $d \in \mathcal{D}$, there are n_d slots. Each slot s corresponds to a specific aspect of the user intent (e.g. *price*) and can take a value v (e.g. *cheap*) from a candidate value set defined by a domain ontology $\mathcal{O}$. The dialogue state S can be defined as a set of (d, s, v) triples, e.g. { (*hotel, price-range, cheap*), (*restaurant, area, west*)}.

At t-th dialogue turn, the dialogue state is S_t, which is used as a constraint to frame a database query. Based on the results of database query and S_t, the system gives a response R_{t+1} to the user. Then the user inputs a new sentence U_{t+1}. A state tracker then updates the dialogue state from S_t to S_{t+1} according to R_{t+1} and U_{t+1}. The whole dialogue process can be represented as a set of triples $\{(S_0, R_1, U_1), (S_1, R_2, U_2), \cdots, (S_{T-1}, R_T, U_T)\}$.

Traditionally, lots of DST models predict dialogue state according to the whole dialogue context up to date. They do not explicitly model the dialogue

state update process. In contrast, our proposed model explicitly updates dialogue state from S_t to S_{t+1} depending on R_{t+1} and U_{t+1},

$$S_{t+1} = f_{dst}(S_t, R_{t+1}, U_{t+1}). \tag{1}$$

Concretely, our proposed DST model takes the triple (S_t, U_{t+1}, R_{t+1}) as input, and predict a distribution over all candidate values for each slot.

As shown in Figure 2, our proposed model consists of four components: input encoding module, context-aware slot embedding, multi-hop gated memory network and slot gate classifier.

3.1 Input Encoding Module

The input encoding module takes the previous dialogue state S_{t-1} and the current dialogue utterances R_t and U_t as input, and output the representation for each token in the input.

As mentioned above, the dialogue state S_{t-1} is a set of domain-slot-value triples. Therefore, we denote the previous state as $S_{t-1} = Z^1_{t-1} \oplus Z^2_{t-1} \oplus \cdots \oplus Z^J_{t-1}$, where J is the number of triples in the dialogue state. Each triple Z^i_{t-1} is a denoted as a sub-sequence, i.e. $Z^i_{t-1} = d \oplus s \oplus v$, e.g. {(*hotel, price-range, cheap*), (*restaurant, area, west*)} is translated to a sequence "*hotel price range cheap restaurant area west*". The state sequence is then contacted with the dialogue utterances as the input of the encoder. Here, we utilize the deep contextual pre-trained language model BERT [6] as the encoder:

$$\mathbf{H}_t = \text{BERT}\left([\text{CLS}] \oplus S_{t-1} \oplus [\text{SEP}] \oplus R_t \oplus U_t \oplus [\text{SEP}]\right), \tag{2}$$

where [CLS] and [SEP] are special tokens for separation in BERT. $\mathbf{H}_t = \{\mathbf{h}^k_t\}^N_{k=1}$, where $\mathbf{h}^k_t$ is the representation vector of k-th token in the joint input sequence and N is the number of tokens in the input sequence [3].

3.2 Context-aware Slot Embedding

The input sequence described in Section 3.1 usually contains information of more than one slot. For j-th (domain, slot) pair, we need to harvest its related information. First, we use the pre-trained BERT to encode the domain and slot tokens, and take the corresponding representation vector of "[CLS]" as the initial slot embedding $\mathbf{s}_j$. Then, we obtain the context-aware slot representation $\mathbf{c}_j$ according to the attention mechanism,

$$\mathbf{c}_j = \sum_{k=1}^{N} \alpha^k \mathbf{h}^k, \tag{3}$$

$$\alpha^k = \text{softmax}\left(\mathbf{s}_j^T \left(\mathbf{W}_{att}\mathbf{h}^k + \mathbf{b}_{att}\right)\right), \tag{4}$$

[3] For brevity, the subscript t of $\mathbf{h}^k_t$ will be omitted in the following sections.

where $\mathbf{W}_{att} \in \mathcal{R}^{d \times d}$ and $\mathbf{b}_{att} \in \mathcal{R}^{d}$ are trainable parameters, d is the dimension of vectors, and $\mathbf{s}_j$ is also updated during training.

This context-aware slot embedding $\mathbf{c}_j$ can be regarded as a summary of dialogue context in the view of the j-th (domain, slot) pair, and it can be used as the query vector to retrieve the related values in the ontology memory.

3.3 Multi-hop Gated Memory Network

For the j-th (domain, slot) pair [4], we utilize the multi-hop gated memory network (MH-GMN) to find the related value according to the context-aware slot embedding $\mathbf{c}_j$. MH-GMN consists of multi-layer supporting memories, each of which is in turn comprised of a set of input and output memory representations with memory cells. At the l-th layer, the input and output memory cells are obtained by transforming the candidate values $\{v_1, \cdots, v_K\}$ in ontology using two trainable embeddings $\mathbf{A}^l \in \mathcal{R}^{d \times K}$ and $\mathbf{E}^l \in \mathcal{R}^{d \times K}$. For both embeddings, their i-th column vectors can be initialized with the contextual representation using BERT, i.e. the domain, slot and i-th candidate value v_i as well as two tokens "[CLS]" and "[SEP]" are concatenated as the input of BERT:

$$\mathbf{h}_i^{[\mathrm{CLS}]} = \mathrm{BERT}([\mathrm{CLS}] \oplus \mathrm{domain} \oplus \mathrm{slot} \oplus v_i \oplus [\mathrm{SEP}]), \tag{5}$$

where $\mathbf{h}_i^{[\mathrm{CLS}]}$ is the contextual representation of "[CLS]" and it is used as the initial embedding vector [5].

MH-GMN takes $\mathbf{c}_j$ as the initial query vector $\mathbf{q}^0$, and updates it from one hop to the next. At the l-th hop, we use $\mathbf{q}^l$ to compute dot product attention scores $\mathbf{p}_i^l (1 \leq i \leq K)$ over each entry of the input memory cells,

$$p_i^l = \mathrm{softmax}((\mathbf{q}^l)^T \cdot \mathrm{a}_i^l), \tag{6}$$

where $\mathbf{a}_i^l$ is the i-th column vector of the input embedding matrix $\mathbf{A}^l$, i.e. it is the input embedding vector of the i-th candidate value v_i. Subsequently, we calculate the output memory $\mathbf{o}^l$ by applying weighted sum of attention scores and the output memory cells:

$$\mathbf{o}^l = \sum_{i=1}^{K} p_i^l \mathrm{e}_i^l, \tag{7}$$

where $\mathbf{e}_i^l$ is the i-th column vector of the output embedding matrix $\mathbf{E}^l$.

We use end-to-end memory access regulation mechanism in the updating procedure. We define the forget gate as:

$$\mathbf{g}^l = \sigma(\mathbf{W}_g^l \mathbf{q}^l + \mathbf{b}_g^l), \tag{8}$$

[4] For brevity, the subscript indicating the (domain, slot) pair is omitted in this section and next section.

[5] When the size of embedding vector and the size of BERT embedding are different, a linear transformation layer will be used.

where $\mathbf{W}_g^l \in \mathcal{R}^{d\times d}$ and $\mathbf{b}_g^l \in \mathcal{R}^d$ are the hop-specific trainable parameters for the l-th hop and σ is the sigmoid function. The updating rule of the query vector is defined by:

$$\mathbf{q}^{l+1} = \mathbf{o}^l \odot (1 - \mathbf{g}^l) + \mathbf{q}^l \odot \mathbf{g}^l, \tag{9}$$

where $\odot$ denotes the element-wise product of two vectors.

Finally, we take the attention scores $p_i^L (1 \leq i \leq K)$ of the last hop as the slot value distribution.

3.4 Slot Gate Classifier

Following the previous work [19], here a slot gate is also utilized to predict the two specific slot values *dontcare* and *none*. We use a context-aware gate classifier to map the context query into three classes {*value, dontcare, none*}. For the j-th (domain, slot) pair, the gate classifier takes the final query vector $\mathbf{q}^L$ as input and predict which class it belongs to:

$$\mathbf{p}_c = \text{softmax}(\mathbf{W}_c \cdot \mathbf{q}^L), \tag{10}$$

where $\mathbf{W}_c \in \mathcal{R}^{3\times d}$ is a parameter for the three-way linear layer.

4 Experimental Setup

4.1 Dataset

We conduct experiments on MultiWoz 2.0 [1] and MultiWoz 2.1 [7], which are among the largest publicly available multi-domain task-oriented dialogue datasets. Both datasets include over 10, 000 dialogues covering seven domains. MultiWoz 2.1 is a renewed version of MultiWoz 2.0 after correcting some false annotations and improper utterance. As is reported by [7], MultiWoz 2.1 corrected over 32% of state annotations across 40% of the dialogue turns and fixed 146 dialogue utterances by canonicalizing slot values in the utterances to the values in the dataset ontology.

Following [19], we use only five domains (*restaurant*, *train*, *hotel*, *taxi*, *attraction*) out of all seven domains since other two domains (*hospital*, *bus*) only make up small portion of the training set and does not even appear in test set. Instead of compacting the whole dialogue utterances, we preprocess the dataset by concatenating the current dialogue state and the utterance of the current turn. In most previous works, the longest input sequence is 879 tokens, while now we only have to process at most 150 tokens, which makes the training process more efficient.

4.2 Training Details

We employed the pretrained BERT-base-uncased for the utterance encoder, memory layer initialization, and domain-slot embedding initialization. The embedding size of memory layers and the domain-slot embedding size is 768, which is predefined in BERT-base-uncased. We use Adam as our optimizer.

Models	Predefined Ontology	BERT used	MultiWoz 2.0	MultiWoz 2.1
HJST [7]	Y	N	38.40	35.55
FJST [7]	Y	N	40.20	38.00
TRADE [19]	N	N	48.60	45.60
Ours (GRU)	Y	N	**49.85**	**51.79**
SOM-DST [12]	N	Y	**51.38**	52.57
DS-DST [20]	Y	Y	-	51.21
DST-picklist [20]	Y	Y	-	53.30
SST [4]	Y	Y	51.17	55.23
Trippy [11]	N	Y	-	**55.29**
Ours (BERT)	Y	Y	50.15	52.70

Table 1. Main results of our approaches and several baselines on the test set of MultiWoz 2.0 and MultiWoz 2.1,

For memory layer initialization and domain-slot embedding initialization, we use the BERT output corresponding to [CLS] token. For example, we use the output $\mathbf{h}^{[\mathrm{CLS}]}$ of "[CLS] hotel price range cheap [SEP]" as a representation for the triplet "hotel-pricerange-cheap".

We use different learning rates for utterance encoder and memory networks. We set the learning rate to 1e-5 for utterance encoder and 1e-4 for memory networks. We use a batch size of 16 and set the dropout rate to 0.2. The max sequence length is fixed to 256.

During training, we regard the model as a branched multi-layer architecture, where each domain-slot subtask shares the encoder parameters and has a corresponding unique memory network. Since we use adjacent architecture in Memory Network implementation, there are $L + 1$ different layers for a L-hop memory network. To prevent overfitting, we sequentially freeze all but one layer and only train that layer for one epoch. We use teacher forcing to train the memory network.

In our model, dialogue context is encoded with BERT [6], memory embeddings and domain-slot embeddings are also initialized with BERT. To make a fair comparison with previous models, we also conducted experiments by replacing BERT with BiGRU [5], and we initialize the utterance word embedding, memory embeddings, and domain-slot embedding by concatenating Glove embeddings [15] and character embeddings [10], where the dimension is 400 for each vocabulary word.

5 Experimental Results

5.1 Baseline Models

TRADE encodes the dialogue context and decodes the value for every slot using a copy mechanism in dialogue state sequence generation. It is also capable of transferring to unseen domains [19].

DST-picklist is proposed together with DS-DST, but this model assumed that the full ontology is available and only performed picklist-based DST [20].

SST predicts dialogue statesfrom dialogue utterances and schema graphs which containslot relations in edges, It uses a graph attention matching network to fuse information from utterances and graphs, and a recurrent graph attention network to control state updating. [4].

Trippy makes use of various copy mechanisms to fill slots with values, it combines the advantages of span-based slot filling methods with memory methods to avoid the use of value picklists altogether [11].

5.2 Joint Goal Accuracy

We compare our approach with several baselines. For performance evaluation, we use joint goal accuracy, an evaluation metric that checks whether the predicted values of all slots exactly match those of the ground truth. The experimental results on the test sets of MultiWoz 2.1 and MultiWoz 2.0 are reported in Table 1. As we can see, our model achieves 50.15% joint accuracy on MultiWoz 2.0 and 52.70% on MultiWoz 2.1, which is compatible among models using BERT. To make a fair comparison with previous non-BERT models, we conducted experiments using GRU instead of BERT. The non-BERT version achieves 49.85% and 52.70% joint accuracy respectively on MultiWoz 2.0 and MultiWoz 2.1, which proves the effectiveness of our model among previous non-BERT models. Interestingly, on the contrary to most previous works, our model achieves higher performance on MultiWoz 2.1 than on MultiWoz 2.0. This phenomenon is consistent with the report by SOM-DST [12]. As they assumed, models explicitly using the dialogue state labels as input, like SOM-DST and our model, benefit more from the error correction on the state annotations done in MultiWOZ 2.1.

5.3 Slot-Specific Accuracy

Figure 3 shows the comparison between the slot-specific accuracy of TRADE and the non-BERT version of our model on the test set of MultiWoz 2.1. We can conclude that our model achieves much better performance than the TRADE model with regard to almost all sub-tasks. Specifically, our model outperforms TRADE by more than 2% accuracy under *train-leaveat*, *restaurant-name*, *taxi-destination* and *taxi-departure*, where many possible values are provided. We can conclude that our model shows robustness even when confronted with many distractors.

5.4 Ablation Study on Memory Gates

We conducted an ablation study on the use of Memory Gates, the result is shown in Figure 4. As can be seen from the table, generally using Memory Gates

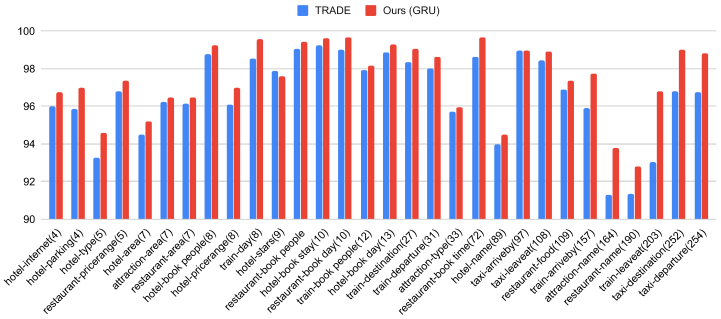

Fig. 3. Slot-Specific Accuracy of TRADE and the RNN version of our model on the test set of MultiWoz 2.1. The numbers in brackets indicates how many possible values there are under the domain-slot pairs.

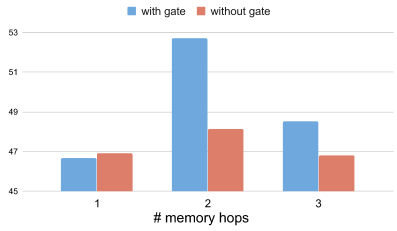

Fig. 4. Joint Accuracy variation with and without gates using different number of memory hops

benefits the performance. With the two-hops memory network, using Memory Gates achieves over 4% performance gain in joint accuracy. Interestingly, under the one-hop memory network, the use of Memory Gates poses only a slight impact over the performance. This is presumably because that for a one-hop memory network, only one Memory Gate is used, which gains only little benefit from the memory updating mechanism.

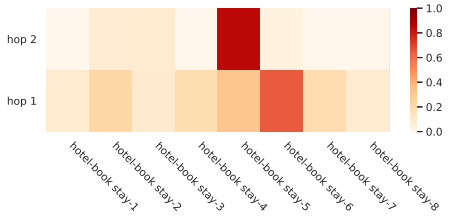

Fig. 5. Attention scores taken from different hops related to <hotel-book stay>. The turn utterance is 'i would recommend express by holiday inn cambridge . from what day should i book ? ; starting saturday . i need 5 nights for 6 people by the way .'

5.5 Ablation Study on number of Memory Hops

We conducted experiments with different numbers of memory hops, the result is shown in Figure 4. As can be suggested, with or without memory gates, our model achieves the best performance using two memory hops. Interestingly, the performance of our model drops when the number of memory hops increase, which is contradictory to intuition. This is presumably because that dialogue information is blurred due to query updating. When 4 hops of memory layer are used, the performance of our model drops dramatically, indicating the equal importance of making complex reasoning and properly understanding dialogue information.

Figure 5 presents an example for showing how memory networks can be powerful in making complex reasoning. This example is taken from the test set of MultiWoz 2.1, involving a customer trying to book a hotel room. At the first hop, memory network seems misled by the information that our customer wants to book for 6 people. While with the help of the query updating mechanism, the model can correct itself at the second hop.

6 Conclusion

In this work, we present a novel model for multi-domain dialogue state tracking, which combines Gated Memory Network and pre-trained BERT [6] to increase

the ability of complex reasoning. We also conducted experiments over RNN-based implementation, instead of using BERT as an utterance encoder, to make a fair comparison with former non-BERT models. Experiments show that our approach achieves compatible performance compared with previous approaches and reaches state-of-the-art performance among non-BERT models.

7 Acknowledgement

We thank the anonymous reviewers for their thoughtful comments. This work has been supported by the National Key Research and Development Program of China (Grant No. 2017YFB1002102) and Shanghai Jiao Tong University Scientific and Technological Innovation Funds (YG2020YQ01).

References

1. Budzianowski, P., Wen, T.H., Tseng, B.H., Casanueva, I., Ultes, S., Ramadan, O., Gasic, M.: 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)
2. Chen, H., Liu, X., Yin, D., Tang, J.: A survey on dialogue systems: Recent advances and new frontiers. Acm Sigkdd Explorations Newsletter **19**(2), 25–35 (2017)
3. Chen, L., Chen, Z., Tan, B., Long, S., Gašić, M., Yu, K.: Agentgraph: Toward universal dialogue management with structured deep reinforcement learning. IEEE/ACM Transactions on Audio, Speech, and Language Processing **27**(9), 1378–1391 (2019)
4. Chen, L., Lv, B., Wang, C., Zhu, S., Tan, B., Yu, K.: Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In: AAAI. pp. 7521–7528 (2020)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL (2019)
7. Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S., Hakkani-Tur, D.: Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. arXiv preprint arXiv:1907.01669 (2019)
8. Goddeau, D., Meng, H., Polifroni, J., Seneff, S., Busayapongchai, S.: A form-based dialogue manager for spoken language applications. In: Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96. vol. 2, pp. 701–704. IEEE (1996)
9. Goel, R., Paul, S., Hakkani-Tür, D.: Hyst: A hybrid approach for flexible and accurate dialogue state tracking. arXiv preprint arXiv:1907.00883 (2019)
10. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: Growing a neural network for multiple nlp tasks. arXiv preprint arXiv:1611.01587 (2016)
11. Heck, M., van Niekerk, C., Lubis, N., Geishauser, C., Lin, H.C., Moresi, M., Gašić, M.: Trippy: A triple copy strategy for value independent neural dialog state tracking. arXiv preprint arXiv:2005.02877 (2020)

12. Kim, S., Yang, S., Kim, G., Lee, S.W.: Efficient dialogue state tracking by selectively overwriting memory. arXiv preprint arXiv:1911.03906 (2019)
13. Liu, B., Lane, I.: An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In INTERSPEECH (2017)
14. Paul, S., Goel, R., Hakkani-Tür, D.: Towards universal dialogue act tagging for task-oriented dialogues. arXiv preprint arXiv:1907.03020 (2019)
15. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
16. Perez, J., Liu, F.: Dialog state tracking, a machine reading approach using memory network. arXiv preprint arXiv:1606.04052 (2016)
17. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. In: Advances in neural information processing systems. pp. 2440–2448 (2015)
18. Wen, T.H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L.M., Su, P.H., Ultes, S., Young, S.: A network-based end-to-end trainable task-oriented dialogue system. In EACL (2016)
19. Wu, C.S., Madotto, A., Hosseini-Asl, E., Xiong, C., Socher, R., Fung, P.: Transferable multi-domain state generator for task-oriented dialogue systems. arXiv preprint arXiv:1905.08743 (2019)
20. Zhang, J.G., Hashimoto, K., Wu, C.S., Wan, Y., Yu, P.S., Socher, R., Xiong, C.: Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. arXiv preprint arXiv:1910.03544 (2019)
21. Zhong, V., Xiong, C., Socher, R.: Global-locally self-attentive dialogue state tracker. arXiv preprint arXiv:1805.09655 (2018)