GraphDialog: Integrating Graph Knowledge into End-to-End Task-Oriented Dialogue Systems

Shiquan Yang, Rui Zhang*, Sarah Erfani
The University of Melbourne, Australia
{shiquan@student., rui.zhang@, sarah.erfani@}unimelb.edu.au

Abstract

End-to-end task-oriented dialogue systems aim to generate system responses directly from plain text inputs. There are two challenges for such systems: one is how to effectively incorporate external knowledge bases (KBs) into the learning framework; the other is how to accurately capture the semantics of dialogue history. In this paper, we address these two challenges by exploiting the graph structural information in the knowledge base and in the dependency parsing tree of the dialogue. To effectively leverage the structural information in dialogue history, we propose a new recurrent cell architecture which allows representation learning on graphs. To exploit the relations between entities in KBs, the model combines multi-hop reasoning ability based on the graph structure. Experimental results show that the proposed model achieves consistent improvement over state-of-the-art models on two different task-oriented dialogue datasets.

1 Introduction

Task-oriented dialogue systems aim to help user accomplish specific tasks via natural language interfaces such as restaurant reservation, hotel booking and weather forecast. There are many commercial applications of this kind (e.g. Amazon Alexa, Google Home, and Apple Siri) which make our life more convenient. Figure 1 illustrates such an example where a customer is asking for the information about restaurants. By querying the knowledge base (KB), the agent aims to provide the correct restaurant entities from the KB to satisfy the customer in a natural language form. Hence, the ability to understand the dialogue history, and to retrieve relevant information from the KB is essential in task-oriented dialogue systems.

One approach for designing task-oriented dialogue systems is the pipeline approach (Williams and Young, 2007; Lee et al., 2009; Young et al., 2013), but it suffers from the difficulty in credit assignment and adaption to new domains. Another popular approach is the end-to-end models (Serban et al., 2016; Wen et al., 2017; Williams et al., 2017; Zhao et al., 2017; Serban et al., 2017), which directly map the dialogue history to the output responses. This approach has attracted more attention in the research community recently as it alleviates the drawbacks of the pipeline approach. However, end-to-end dialogue models usually suffer from ineffective use of knowledge bases due to the lack of appropriate framework to handle KB data.

To mitigate this issue, recent end-to-end dialogue studies (Eric et al., 2017; Madotto et al., 2018) employ memory networks (Weston et al., 2015; Sukhbaatar et al., 2015) to support the learning over KB, and have achieved promising results via integrating memory with copy mechanisms (Gulcehre et al., 2016; Eric and Manning, 2017). By using memory, they assume that the underlying structure of KB is linear since memory can be viewed as a

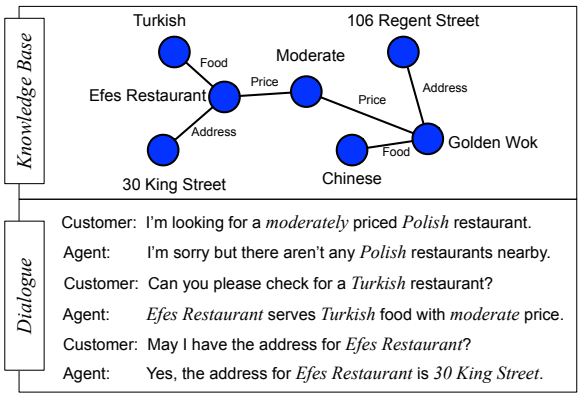


Figure 1: An example dialogue in the restaurant booking domain. The top part is knowledge base (KB) information that represented by a graph and the bottom part is the conversation between a customer and the agent. Our aim is to predict the agent responses given KB information and the customer utterances.

*Rui Zhang is the corresponding author.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 1878–1888,
November 16–20, 2020.

list structure. As a result, the relationships between entities are not captured. However, since KB is naturally a graph structure (nodes are entities and edges are relations between entities). By overlooking such relationships, the model fails to capture substantial information embedded in the KB including the semantics of the entities which may significantly impact the accuracy of results. Moreover, structural knowledge such as dependency relationships has recently been investigated on some tasks (e.g., relation extraction) (Peng et al., 2017; Song et al., 2018) and shown to be effective in the model's generalizability. However, such dependency relationships (essentially also graph structure) have not been explored in dialogue systems, again missing great potential for improvements.

With the above insight, we propose a novel **graph**-based end-to-end task-oriented **dialog**ue model (**GraphDialog**) aimed to exploit the graph knowledge both in dialogue history and KBs. Unlike traditional RNNs such as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014), we design a novel recurrent unit (Section 3.1.2) that allows multiple hidden states as inputs at each timestep such that the dialogue history can be encoded with graph structural information. The recurrent unit employs a masked attention mechanism to enable variable input hidden states at each timestep. Moreover, We incorporate a graph structure (Section 3.2) to handle the external KB information and perform multi-hop reasoning on the graph to retrieve KB entities.

Overall, the contributions of this paper are summarized as follows:

- We propose a novel graph-based end-to-end dialogue model for effectively incorporating the external knowledge bases into task-oriented dialogue systems.
- We further propose a novel recurrent cell architecture to exploit the graph structural information in the dialogue history. We also combine the multi-hop reasoning ability with graph to exploit the relationships between entities in the KB.
- We evaluate the proposed model on two real-world task-oriented dialogue datasets (i.e., SMD and MultiWOZ 2.1). The results show that our model outperforms the state-of-the-art models consistently.

2 Related Work

Task-oriented dialogue system has been a long-standing studied topic (Williams and Young, 2007; Lee et al., 2009; Huang et al., 2020b) and can be integrated into many practical applications such as virtual assistant (Sun et al., 2016, 2017). Traditionally, task-oriented dialogue systems are built in the pipeline approach, which consists of four essential components: natural language understanding (Chen et al., 2016), dialogue state tracking (Lee and Stent, 2016; Zhong et al., 2018; Wu et al., 2019a), policy learning (Su et al., 2016; Peng et al., 2018; Su et al., 2018) and natural language generation (Sharma et al., 2017; Chen et al., 2019; Huang et al., 2020a). Another recent approach is the end-to-end models (Wu et al., 2018; Lei et al., 2018), which directly map the user utterances to responses without heavy annotations. Bordes et al. (2017) apply end-to-end memory networks (Sukhbaatar et al., 2015) for task-oriented dialogues and shown that end-to-end models are promising on the tasks. To produce more flexible responses, several generative models are proposed (Zhao et al., 2017; Serban et al., 2016). They formulate the response generation problem as a translation task and apply sequence-to-sequence (Seq2Seq) models to generate responses. Seq2Seq models have shown to be effective in language modeling but they struggle to incorporate external KB into responses. To mitigate this issue, Eric and Manning (2017) has enhanced the Seq2Seq model by adding copy mechanism. Madotto et al. (2018) combines the idea of pointer with memory networks and obtained improved performance. Wu et al. (2019b) incorporates global pointer mechanism and achieved improved performance. Our study differs from those works in that we exploit the powerful graph information both contained in the dialogue history and in the KBs to effectively incorporate KBs into dialogue systems.

3 Proposed Model

Our proposed model consists of three components: an encoder (Section 3.1), a decoder (Section 3.3) and a knowledge graph with multi-hop reasoning ability (Section 3.2). Formally, let $X = \{x_1,...,x_n\}$ be a sequence of tokens, where each token $x_i \in X$ corresponds to a word in the dialogue history. We first obtain a *dialogue graph* $\widehat{G}$ (Section 3.1.1), which is the dependency parsing graph of the sentences in the dialogue history X, as the input of the encoder. The encoder then learns a fixed-length

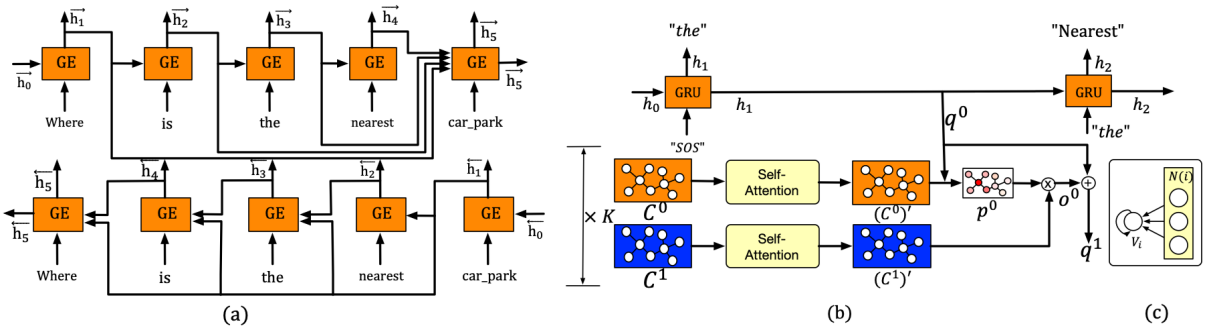


Figure 2: Overview of the proposed architecture. (a) Graph Encoder, top is forward graph and bottom is backward graph. (b) Decoder and Knowledge Graph with multi-hop reasoning mechanism. (c) Self-Attention Mechanism.

vector as the encoding of the dialogue history based on $\widehat{G}$, which is then fed to the decoder for hidden state initialization. The knowledge graph adopts another graph $G = \{V,E\}$ to store and retrieve the external knowledge data (Section 3.2.1), where V denotes the entities and E denotes the edges. The decoder generates the system response $Y = \{y_1,...,y_m\}$ token-by-token either by copying entities from graph G via querying the knowledge graph or by generating tokens from vocabularies. Figure 2 illustrates the overall architecture of the proposed model. In the following sections, we describe each component in detail.

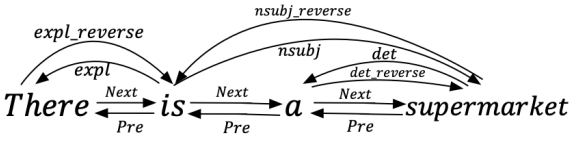


Figure 3: An example of dialogue graph.

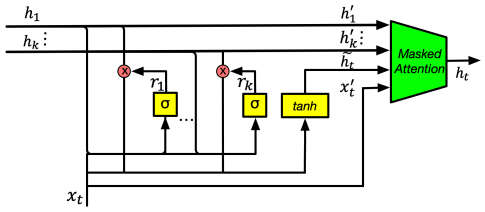


Figure 4: Overview of the proposed recurrent unit.

3.1 Graph Encoder

3.1.1 Dialogue Graph

To enable learning semantic rich representations of words with various relationships, such as adjacency and dependency relations, we first use the off-the-shelf tool *spacy*[1] to extract the dependency relations among the words in the dialogue history X. Figure 3 gives an example of the dependency parsing result. The bi-directional edges among words allow information flow both from dependents to heads and from heads to dependents. The intuition is that the representation learning of the head words should be allowed being influenced by the dependent words and vice versa, thus allowing the learning process to capture the mutual relationships between the head words and the dependent words to provide richer representation.

We compose the dialogue graph by combining the obtained dependency relations with the sequential relations (i.e., *Next* and *Pre*) among words, which serves as the input to the graph encoder. To further support bi-directional representation learning, we split the obtained dialogue graph into two parts: the *forward graph* (from left to right) and the *backward graph* (from right to left).

[1] https://spacy.io/

3.1.2 Recurrent Cell Architecture

The recurrent cell architecture (Figure 4) is the core computing unit of the graph encoder, and is used to compute the hidden state of each word in the obtained dialogue graph. The cell traverse the words in the dialogue graph sequentially according to the word order in the dialogue history. Next, we show how to compute the cell hidden state h_t at timestep t.

Let us define x_t as the input word representation at timestep t. $P(t) = \{p_1,p_2,\ldots,p_k\}$ is the set of precedent words for x_t where each $p_i \in P(t)$ denotes a word in the dialogue graph that connects to x_t, and k is the total number of the precedents of x_t. $H = \{h_1,h_2,\ldots,h_k\}$ is a set of hidden states where each element $h_j \in H$ denotes the hidden state of the j-th predecessor $p_j \in P(t)$.

The input of the cell consists of two parts: the input word vector x_t, and the predecessor hidden states H. First, we loop over the k hidden states

in H and compute a *reset gate* for each of them. Specifically, we compute r_j for the *j*-th hidden state using:

$$r_j = \sigma\left(W_r x_t + U_r h_j\right) \tag{1}$$

where σ is the logistic sigmoid function, x_t and h_j are the current input and the hidden state of the *j*-th predecessor at timestep *t* respectively. W_r and U_r are parameters which will be learned. We then compute a candidate hidden state $\widetilde{h}_t$ using:

$$\widetilde{h}_t = \phi\left(W_n x_t + \frac{1}{k}\sum_{j=1}^{k} r_j * (U_n h_j)\right) \tag{2}$$

where ϕ is the hyperbolic tangent function, *k* is the number of predecessors of word x_t, W_n and U_n are the learnable weight matrices. Intuitively, $\widetilde{h}_t$ is the contextualized representation of current input x_t.

Next, we combine the obtained candidate hidden state $\widetilde{h}_t$ with the predecessor hidden states H, and use an *masked attention mechanism* (Equation 6) to aggregate them together to yield the output hidden state h_t at timestep *t*. To obtain sufficient expressive power, we first apply linear transformations to the input x_t and the hidden states $h_j \in H$ using:

$$x_t^{'} = W_z x_t \tag{3}$$

$$h_j^{'} = U_z h_j \tag{4}$$

where W_z, U_z are parameters which are learned, t is the current timestep. We denote $H^{'}$=$\{h_1^{'},h_2^{'},\ldots,h_k^{'}\}$ as the transformed set of hidden states. Then we add the previously obtained candidate hidden state $\widetilde{h}_t$ into the transformed set of hidden states $H^{'}$ and obtain $H^{''}$=$\{h_1^{'},h_2^{'},\ldots,h_k^{'},\widetilde{h}_t\}$. The intuition is that the output hidden state depends on both the history information ($h_1^{'}$ to $h_k^{'}$) and the current input ($\widetilde{h}_t$).

Then we perform attention mechanism by using the hidden states $H^{''}$ as keys and the current input x_t as query. Intuitively, different inputs (e.g. different predecessors in $H^{''}$) should have different impacts on the output hidden state h_t, and we expect our model to capture that. However, the inputs may have different number of predecessors at different timesteps. To handle this, inspired by (Vaswani et al., 2017), we employ an *masked attention mechanism* to learn the importance of each predecessor at every timestep, thus avoiding the pad information affecting the learning process. We compute the attention using:

$$e_j = v^T \phi\left(x_t^{'} + h_j^{'}\right) \tag{5}$$

$$\alpha_j = \textit{softmax}\left([e_j]_m\right) \tag{6}$$

where v is a learnable parameter, $h_j^{'}$ is the *j*-th vector in $H^{''}$, $softmax(z_i)$=$e^{z_i}/\sum_j e^{z_j}$, α_j denotes the attention weight on the *j*-th vector in $H^{''}$, $[\cdot]_m$ denotes the mask operation. In our implementation, we simply set the number to negative infinity if the *j*-th hidden state corresponds to a pad token. Finally, we compute the weighted sum to obtain the cell output hidden state h_t at timestep *t* using:

$$h_t = \sum_{j=1}^{k+1} \alpha_j h_j^{'} \tag{7}$$

Intuitively, the *reset gate* controls the information flow from the multiple predecessors to the hidden state of current timestep. If a precedent word is more correlated to the current input word, then it is expected to let the information of the precedent word flow through the gate to affect the representation of current timestep.

3.1.3 Bi-directional Representation

To obtain a bi-directional representation for the dialogue history, we use the same cell architecture (Section 3.1.2) to loop over the *forward graph* and *backward graph* separately, and compute a forward representation $\overrightarrow{h}_n$ and a backward representation $\overleftarrow{h}_n$, respectively. Then we concatenate them together to serve as the final representation of dialogue history h_n^e=$[\overrightarrow{h}_n;\overleftarrow{h}_n]$, which will become a part of the inputs to the decoder.

3.2 Multi-hop Reasoning Mechanism over Knowledge Graph

A straightforward way to explore the graph information in KB is to represent the KB as a graph structure, and then query the graph using attention mechanism with the decoder hidden states. However, our preliminary experiments didn't show a good performance using this approach. We conjecture that it may be due to the poor reasoning ability of this method. To address this issue, we extend the graph with multi-hop reasoning mechanism, which aimed to strengthen the reasoning ability over graph as well as to capture the graph structural information between entities via self-attention. We call it knowledge graph module in the following sections.

Formally, the knowledge graph module contains two sets of trainable parameters $C = \{C^1, C^2, \ldots, C^{K+1}\}$, where each C^k is an embedding matrix that maps tokens to vector representations, and $V = \{V^1, V^2, \ldots, V^{K+1}\}$, where each V^k is a weight vector for computing self-attention coefficients, and K is the maximum number of hops.

Now we describe how to compute the output vector of the knowledge graph. The model loops over K hops on an input graph. At each hop k, a query vector q^k is employed as the reading head. First, the model uses an embedding layer C^k to obtain the continuous vector representations of each node i in the graph as C_i^k, where $C_i^k = C^k(n_i)$ and n_i is the i-th node in the graph. Then we perform *self-attention mechanism* on the nodes and compute the attention coefficients using:

$$e_{ij} = \varphi\left(\left(V^k\right)^T [C_i^k || C_j^k]\right) \tag{8}$$

where φ is the LeakyReLU activation function (with negative input slope $\alpha = 0.2$), V^k is the parametrized weight vector of the attention mechanism at hop k, C_i^k and C_j^k are the node vectors for the i-th and j-th node in the graph at hop k, and $\|$ is the concatenation operation. We then normalize the coefficients of each node i with respect to all its first-order neighbors using the softmax function:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})} \tag{9}$$

where N_i is the first-order neighbors of node i (including i), *exp* is the exponential function.

Then we update the representation of each node i by a weighted sum of its neighbors in N_i using:

$$\left(C_i^k\right)' = \sum_{j \in N_i} \alpha_{ij} C_j^k \tag{10}$$

Next, the query vector q^k is used to attend to the updated nodes in the graph and compute the attention weights for each node i at hop k using:

$$p_i^k = softmax\left(\left(q^k\right)^T \left(C_i^k\right)'\right) \tag{11}$$

To obtain the output of the knowledge graph, we apply the same self-attention mechanism (Equations 8 and 9) and update strategy (Equation 10) to the node representation C_i^{k+1}. We use C^{k+1} here since the adjacent weighted tying strategy is adopted. The updated node representation for output is denoted as $\left(C_i^{k+1}\right)'$. Once obtained, the model reads out the graph o^k by the weighted sum over it using:

$$o^k = \sum_i p_i^k \left(C_i^{k+1}\right)' \tag{12}$$

Then the query vector q^k is updated for the next hop using $q^{k+1} = q^k + o^k$. The final output of the knowledge graph is o^K, which will become a part of the inputs to the decoder.

3.2.1 Graph Construction

In practice, dialogue systems usually use KBs (mostly in a relational database format) to provide external knowledge. We have converted the original relational database into a graph structure to exploit the relation information between KB entities. First, we find all the entities in the relational database as the nodes of the graph. Then we assign an edge to a pair of entities if there exists relationship between them according to the records in the relational database. Thus we can obtain the graph structured external knowledge.

3.3 Decoder

We use a standard Gated Recurrent Unit (GRU) (Cho et al., 2014) as the decoder to generate the system response word-by-word. The initial hidden state h_0 consists of two parts: the graph encoder output and the knowledge graph output. We take the output hidden state of the graph encoder h_n^e as the initial query vector q^0 to attend to the knowledge graph and obtain the output o^K. The initial hidden state h_0 is then computed using:

$$h_0 = [h_n^e || o^K] \tag{13}$$

At each decoder timestep t, the GRU takes the previously generated word $\widehat{y}_{t-1}$ and the previous hidden state h_{t-1} as the input and generates a new hidden state h_t using:

$$h_t = GRU\left(\widehat{y}_{t-1}, h_{t-1}\right) \tag{14}$$

Next, we follow (Wu et al., 2019b) that the decoder learns to generate a sketch response that the entities in the response are replaced with certain tags. The tags are obtained from the provided ontologies in the training data. The hidden state h_t are used for two purposes. The first one is to generate a vocabulary distribution P_{vocab} over all the words in the vocabulary using:

$$P_{vocab} = softmax\left(W_o h_t\right) \tag{15}$$

where W_o is the learnable parameter. The second one is to query the knowledge graph to generate a graph distribution P_{graph} over all the nodes in the graph. We use the attention weights at the last hop of the knowledge graph p_t^K as P_{graph}.

At each timestep t, if the generated word from P_{vocab} (the word has the maximum posterior probability) is a tag, then the decoder choose to copy from the graph entities that has the largest attention value according to P_{graph}. Otherwise, the decoder will generate the target word from P_{vocab}. During training, all the parameters are jointly learned via minimizing the sum of two cross-entropy losses: one is between P_{vocab} and $y_t \in Y$, and the other is between P_{graph} and G_t^{Label}, where G_t^{Label} is the node id that corresponds to the current output y_t.

4 Experiments

4.1 Dataset

To validate the efficacy of our proposed model, we evaluate it on two public multi-turn task-oriented diaglogue datasets: Stanford multi-domain dialogue (SMD) (Eric et al., 2017) and MultiWOZ 2.1 (Eric et al., 2019). The SMD is a human–human dataset for in-car navigation task. It includes three distinct task domains: *point-of-interest navigation*, *calendar scheduling* and *weather information retrieval*. The MultiWOZ 2.1 dataset is a recently released human–human dialogue corpus with much larger data size and richer linguistic expressions that make it a more challenging benchmark for end-to-end task-oriented dialogue modeling. It consists of seven distinct task domains: *restaurant*, *hotel*, *attraction*, *train*, *hospital*, *taxi* and *police*. We select four domains (*restaurant*, *hotel*, *attraction*, *train*) to test our model since the other three domains (*police*, *taxi*, *hospital*) lack KB information which is essential to our task. We will make our code and data publicly available for further study. To the best of our knowledge, we are the first to evaluate end-to-end task-oriented dialogue models on MultiWOZ 2.1. The train/validation/test sets of these two datasets are split in advance by the providers.

4.2 Training Details

We implement our model[2] in Tensorflow and is trained on NVIDIA GeForce RTX 2080 Ti. We use grid search to find the best hyper-parameters for our model over the validation set (use BLEU as criterion for both datasets). We randomly initialize all the embeddings in our implementation. The embedding size is selected between [16,512], which is also equivalent to the RNN hidden state (including the encoder and the decoder). We also use dropout for regularization on both the encoder and the decoder to avoid over-fitting and the dropout rate is set between [0.1,0.5]. We use Adam optimizer (Kingma and Ba, 2015) to accelerate the convergence with a learning rate chosen between [$1e^{-3}$,$1e^{-4}$]. We simply use a greedy strategy to search for the target word in the decoder without advanced techniques like beam-search.

[2]Code and data are available at: https://github.com/shiquanyang/GraphDialog

Metrics	SMD	MultiWOZ 2.1
Avg. Turns per dialog	5.25	13.46
Avg. Tokens per turn	8.02	13.13
Total number of turns	12732	113556
Vocabulary	1601	23689
Train dialogs	2425	8438
Val dialogs	302	1000
Test dialogs	304	1000

Table 1: Dataset statistics for SMD and MultiWOZ 2.1.

4.3 Evaluation Metrics

We use two common evaluation metrics in dialogue studies including BLEU (Papineni et al., 2002) (using Moses `multi-bleu.perl` script) and Entity F1 (Eric et al., 2017; Madotto et al., 2018) for evaluations.

4.4 Effect of Models

We compare our model with several existing models: standard sequence-to-sequence (Seq2Seq) models with and without attention (Luong et al., 2015), pointer to unknown (Ptr-Unk, (Gulcehre et al., 2016)), GraphLSTM (Peng et al., 2017), BERT (Devlin et al., 2019), Mem2Seq (Madotto et al., 2018) and GLMP (Wu et al., 2019b). Note that the results we listed in Table 2 for GLMP is different from the original paper, since we re-implement their model in Tensorflow according to their released Pytorch code for fair comparison.

Stanford Multi-domain Dialogue. Table 2 has shown the results on SMD dataset. Our proposed model achieves a consistent improvement over all the baselines with the highest BLEU score 13.6 and 57.4% entity F1 score. The performance gain in BLEU score suggests that the generation error in the decoder has been reduced. The improvement on entity F1 indicates that our model can retrieve

Model	S2S	S2S + Attn	Ptr-Unk	GraphLSTM	BERT	Mem2Seq	GLMP	GraphDialog		
								K=1	K=3	K=6
BLEU	8.4	9.3	8.3	10.3	9.13	12.6	12.2	12.96	**13.66**	12.74
Entity F1	10.3	19.9	22.7	50.8	49.6	33.4	55.1	56.14	**57.42**	55.90
Schedule F1	9.7	23.4	26.9	69.9	57.4	49.3	67.3	70.96	**71.90**	71.84
Weather F1	14.1	25.6	26.7	46.6	47.5	32.8	54.1	56.89	**59.68**	54.36
Navigation F1	7.0	10.8	14.9	43.2	46.8	20.0	48.4	48.37	**48.58**	47.55

Table 2: Evaluation on SMD dataset. Human, rule-based and KV Retrieval Net results are reported from (Eric et al., 2017), which are not directly comparable since the problem is simplified to canonicalized forms. K denotes the maximum number of hops for knowledge graph. Ours achieves highest BLEU and entity F1 score over baselines.

Model	S2S	S2S + Attn	Ptr-Unk	GraphLSTM	BERT	Mem2Seq	GLMP	GraphDialog		
								K=1	K=3	K=6
BLEU	2.5	3.0	2.3	3.4	3.9	4.1	4.3	5.47	**6.17**	5.14
Entity F1	1.3	2.1	2.5	4.7	4.1	3.2	6.7	9.56	**11.28**	8.74
Restaurant F1	1.6	2.2	2.3	9.8	7.3	2.9	11.4	15.27	**15.95**	13.25
Hotel F1	1.5	3.4	3.8	2.1	1.6	4.5	3.9	7.54	**10.79**	7.05
Attraction F1	0.8	1.4	1.7	7.2	8.4	2.1	9.4	5.78	**14.12**	7.89
Travel F1	0.2	0.7	0.9	1.8	2.1	1.5	3.5	3.41	**4.39**	3.53

Table 3: Evaluation on MultiWOZ 2.1 dataset. Ours achieves highest BLEU and entity F1 score over baselines.

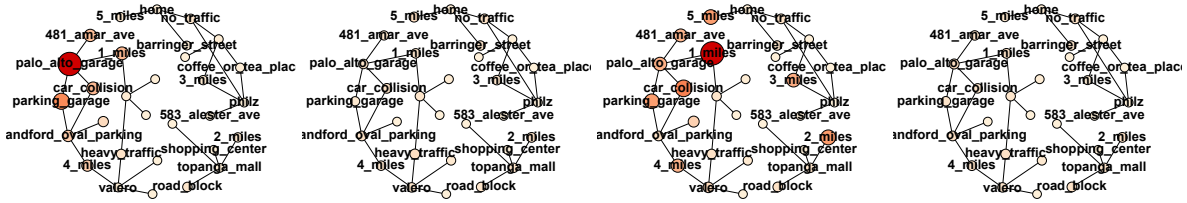
entities from the external knowledge data more accurately than those baselines. We also conduct comparisons with BERT to validate the effectiveness of our proposed model. Specifically, we use the bert-base-uncased model (due to GPU memory limit) from huggingface library[3] as our encoder to encode the dialogue history and the remaining parts are the same as our model. We then fine-tune BERT on our dialogue dataset. We can find that our mode significantly outperforms the fine-tuned BERT by a large margin which further demonstrates the effectiveness of our proposed model. We conjecture that the reasons may lie in two aspects. First, the context of the corpus used for pretraining BERT differs from our dialogue dataset. Secondly, the model complexity of BERT may cause overfitting issue on small-scale datasets like SMD etc.

MultiWOZ 2.1. Table 3 shows the results on a more complex dataset MultiWOZ 2.1. Our model outperforms all the other baselines by a large margin both in entity F1 and BLEU score, which confirms our model has a better generalization ability than those baselines. One may find that the entity F1 and BLEU score has a huge gap between MultiWOZ 2.1 and SMD. This performance degradation phenomenon has also been observed by other dialogue works (Budzianowski et al., 2018) which implies that the MultiWOZ corpus is much more challenging than the SMD dataset for dialogue tasks.

Ablation Study. Table 4 shows the contributions of each components in our model. Ours without graph encoder means that we do not use the dependency relations information and the proposed recurrent cell architecture. We simply use a bi-directional GRU to serve as the encoder and the other parts of the model remain unchanged. We can observe that our model without the graph encoder has a 1.6% absolute value loss (over 25% in ratio) in BLEU score and a 1.1% absolute value loss (9.8% in ratio) in entity F1 on MultiWOZ 2.1, which suggests that the overall quality of the generated sentences are better improved by our graph encoder. On the other hand, ours without knowledge graph means that we do not use the graph structure to store and retrieve the external knowledge data. Instead we use memory networks (Sukhbaatar et al., 2015) that has been shown useful to handle the knowledge base similar to (Wu et al., 2019b). We can find a significant entity F1 drop (3.8% in absolute value and 33.9% in ratio) on MultiWOZ 2.1, which verifies the superiority of the proposed graph-based module with multi-hop reasoning ability in retrieving the correct entities, even compared to the strong memory-based baselines.

Model Training Time. We also compare the training time of GraphDialog with those baselines. GraphDialog is about 3 times faster than BERT since its model complexity is smaller. The number

[3] https://github.com/huggingface



(a) Generation *timestep 0* (b) Generation *timestep 1* (c) Generation *timestep 2* (d) Generation *timestep 3*

Figure 5: Knowledge graph attention visualization when generating responses in the SMD navigation domain. Based on the question "Where is a nearby parking_garage?", the generated response of our model is "palo_alto_garage is 1_miles away". Specifically, the attention results at each generation timestep for the knowledge graph information of this example are shown in (a), (b), (c) and (d) respectively. The color and size of the nodes represent their attention weights. The darker and bigger the nodes are, the larger their attention weights are. Our model successfully learns to attend to the correct KB entities (i.e., *palo_alto_garage* and *1_miles* at generation timesteps 0 and 2) which have the highest attention, and the model copies them to serve as the output words. During timesteps 1 and 3, the model generates output words (i.e., *is* and *away*) from the vocabulary.

	SMD		MultiWOZ 2.1	
Model	BLEU	Entity F1(All)	BLEU	Entity F1(All)
GraphDialog	13.66(-)	57.42(-)	6.17(-)	11.28(-)
GraphDialog w/o Graph Encoder	12.35(-1.31)	56.61(-0.81)	4.57(-1.60)	10.13(-1.15)
GraphDialog w/o Knowledge Graph	13.13(-0.53)	55.28(-2.14)	5.35(-0.82)	7.41(-3.87)

Table 4: Model ablation study: Effects of Graph Encoder and Knowledge Graph. Number in the parentheses means the absolute value gap between the full version and the ablation one on corresponding metrics.

	Edge Path Distance			
Dataset	1	≥ 2	≥ 10	≥ 15
SMD	52.82%	33.68%	10.61%	2.89%
MultiWOZ 2.1	50.29%	35.41%	11.26%	3.04%

Table 5: Edge path distance distribution on different datasets.

of parameters for GraphDialog is almost 90% less than BERT, which also saves space for model storage. GraphDialog is slower than GLMP, which is expected as it needs to encode more information. However, the gap of the training time is up to 69%, and we can complete the whole training process within one day which seems reasonable.

4.5 Analysis and Discussion

Why does dependency relations help? We have conducted in-depth analyses from the edge path distance perspective. Table 5 shows the edge path distance distribution in the dialogue graph (Section 3.1.1) on both SMD and MultiWOZ 2.1. The edge path distance is defined as the the number of words between the head word and the tail word along the linear word sequence plus one. For example, for the sentence "There is a supermarket", the edge distance of the "Next" edge between "There" and "is" is 1, the edge path distance of the "nsubj" edge between "is" and "supermarket" is 2. We can find that although many edges have small edge path distances, there are still a considerable number of edges with relatively large distances, which could encourage more direct information flow between distant words in the input. This may partly explain the benefits of using information such as dependency relations in encoding the dialogue history.

Attention Visualization. To further understand the model dynamics, we analyze the attention weights of the knowledge graph module to show its reasoning process. Figure 5 has shown an example of the attention distribution over all the nodes at the last hop of the knowledge graph. Based on the question "Where is a nearby parking_garage?" asked by the user, the generated response of our model is "palo_alto_garage is 1_miles away", and the gold answer is "The nearest one is palo_alto_garage, it's just 1_miles away". We can find that our model has successfully learned to copy the correct entities (i.e., *palo_alto_garage* at timestep 0 and *1_miles* at timestep 2) from the knowledge graph.

Error Analysis. To inspire future improvements, we also inspect the generated responses manually. We find that the model tends to omit

entities when the responses contain multiple KB entities. Besides, about 10% of the generated responses contain duplicate KB entities. For example, "*The temperature in New York on Monday is 100F, 100F*". This may be attributed to the training of GRU in the decoder, and we aim to solve the problem in future work.

5 Conclusion

In this work, we present a novel graph-based end-to-end model for task-oriented dialogue systems. The model leverages the graph structural information in dialogue history via the proposed recurrent cell architecture to capture the semantics of dialogue history. The model further exploits the relationships between entities in the KB to achieve better reasoning ability by combining the multi-hop reasoning ability with graph.

We empirically show that our model outperforms the state-of-the-art models on two real-world task-oriented dialogue datasets. Our model may also be applied to end-to-end open-domain chatbots since the goal is to generate responses given inputs and external knowledge, which is what our model can do. We will explore this direction in future work.

Acknowledgements

We would like to thank Bayu Distiawan Trisedya for his insightful discussions and the valuable feedbacks from all anonymous reviewers. This work is supported by Australian Research Council (ARC) Discovery Project DP180102050.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *5th International Conference on Learning Representations, ICLR 2017*.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 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*.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Yun-Nung Vivian Chen, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *CoRR*, abs/1907.01669.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2020a. MALA: cross-domain dialogue generation with action learning. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'20.

Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2020b. Semi-supervised dialogue policy learning via stochastic reward estimation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Commun.*, 51(5):466–484.

Sungjin Lee and Amanda Stent. 2016. Task lineages: Dialog state tracking for flexible interaction. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2Seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17.

Shikhar Sharma, Jing He, Kaheer Suleman, Hannes Schulz, and Philip Bachman. 2017. Natural language generation in dialogue using lexicalized and delexicalized data. In *5th International Conference on Learning Representations, ICLR 2017, Workshop Track Proceedings*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph-state LSTM. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Online active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. *CoRR*, abs/1808.09442.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*.

Yu Sun, Nicholas Jing Yuan, Yingzi Wang, Xing Xie, Kieran McDonald, and Rui Zhang. 2016. Contextual intent tracking for personal assistants. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. 2017. Collaborative intent prediction with real-time contextual data. *ACM Transactions on Information Systems (TOIS)*, 35(4):1–33.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015*.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient

end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Jason D. Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

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

Chien-Sheng Wu, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2018. End-to-end dynamic query memory network for entity-value independent task-oriented dialog. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Chien-Sheng Wu, Richard Socher, and Caiming Xiong. 2019b. Global-to-local memory pointer networks for task-oriented dialogue. In *7th International Conference on Learning Representations, ICLR 2019*.

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

Tiancheng Zhao, Allen Lu, Kyusong Lee, and Maxine Eskenazi. 2017. Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.