Multi-domain Dialogue State Tracking as Dynamic Knowledge Graph Enhanced Question Answering

Li Zhou
Amazon Alexa Search
lizhouml@amazon.com

Kevin Small
Amazon Alexa Search
smakevin@amazon.com

Abstract

Multi-domain dialogue state tracking (DST) is a critical component for conversational AI systems. The domain ontology (i.e., specification of domains, slots, and values) of a conversational AI system is generally incomplete, making the capability for DST models to generalize to new slots, values, and domains during inference imperative. In this paper, we propose to model multi-domain DST as a question answering problem, referred to as *Dialogue State Tracking via Question Answering* (DSTQA). Within DSTQA, each turn generates a question asking for the value of a (domain, slot) pair, thus making it naturally extensible to unseen domains, slots, and values. Additionally, we use a dynamically-evolving knowledge graph to explicitly learn relationships between (domain, slot) pairs. Our model has a 5.80% and 12.21% relative improvement over the current state-of-the-art model on MultiWOZ 2.0 and MultiWOZ 2.1 datasets, respectively. Additionally, our model consistently outperforms the state-of-the-art model in domain adaptation settings.

1 Introduction

In a task-oriented dialogue system, the dialogue policy determines the next action to perform and next utterance to say based on the current dialogue state. A dialogue state defined by *frame-and-slot semantics* is a set of (key, value) pairs specified by the domain ontology (Jurafsky & Martin, 2019). A key is a (domain, slot) pair and a value is a slot value provided by the user. Figure 1 shows a dialogue and state in three domain contexts. Dialogue state tracking (DST) in multiple domains is a challenging problem. First of all, in production environments, the domain ontology is being continuously updated such that the model must generalize to new values, new slots, or even new domains during inference. Second, the number of slots and values in the training data are usually quite large. For example, the MultiWOZ $2.0/2.1$ datasets (Budzianowski et al., 2018; Eric et al., 2019) have 30 (domain, slot) pairs and more than $4,500$ values (Wu et al., 2019). As the model must understand slot and value paraphrases, it is infeasible to train each slot or value independently. Third, multi-turn inferences are often required as shown in the underlined areas of Figure 1.

Many single-domain DST algorithms have been proposed (Mrkšić et al., 2017; Ren et al., 2018; Zhong et al., 2018). For example, Zhong et al. (2018) learns a local model for each slot and a global model shared by all slots. However, single domain models are difficult to scale to multi-domain settings, leading to the development of multi-domain DST algorithms. For example, Nouri & Hosseini-Asl (2018) improves Zhong et al. (2018)'s work by removing local models and building a slot-conditioned global model to share parameters between domains and slots, thus computing a score for every (domain, slot, value) tuple. This approach remains problematic for settings with a large value set (e.g., *user phone number*). Wu et al. (2019) proposes an encoder-decoder architecture which takes dialogue contexts as source sentences and state annotations as target sentences, but does not explicitly use relationships between domains and slots. For example, if a user booked a restaurant and asks for a taxi, then the destination of the taxi is likely to be that restaurant, and if a user booked

Figure 1: The right-hand side shows a dialogue which involves 3 domains, and the left-hand side shows its dialogue state in the end. Bold text indicate mentions and paraphrases of slot values. Underlined text indicates scenarios where multi-turn inference is required.

a 5 star hotel, then the user is likely looking for an expensive rather than a cheap restaurant. As we will show later, such relationships between domains and slots help improve model performance.

To tackle these challenges, we propose DSTQA (*Dialogue State Tracking via Question Answering)*, a new multi-domain DST model inspired by recently developed reading comprehension and question answering models. Our model reads dialogue contexts to answer a series of questions that asks for the value of a (domain, slot) pair. Specifically, we construct two types of questions: 1) multiple choice questions for (domain, slot) pairs with a limited number of value options and 2) span prediction questions, of which the answers are spans in the contexts, designed for (domain, slot) pairs that have a large or infinite number of value options. Finally, we represent (domain, slot) pairs as a dynamically-evolving knowledge graph with respect to the dialogue context, and utilize this graph to drive improved model performance. Our contributions are as follows: (1) we propose to model multi-domain DST as a question answering problem such that tracking new domains, new slots and new values is simply constructing new questions, (2) we propose using a bidirectional attention (Seo et al., 2017) based model for multi-domain DST, and (3) we extend our algorithm with a dynamically-evolving knowledge graph to further exploit the structure between domains and slots.

2 Problem Formulation

In a multi-domain dialogue state tracking problem, there are M domains $D = \{d_1, d_2, ..., d_M\}$. For example, in MultiWOZ 2.0/2.1 datasets, there are 7 domains: *restaurant*, *hotel*, *train*, *attraction*, *taxi*, *hospital*, and *police*. Each domain $d \in D$ has N^d slots $S^d = \{s_1^d, s_2^d, ..., s_{N^d}^d\}$, and each slot $s \in S^d$ has K^s possible values $V^s = \{v_1^s, v_2^s, ..., v_{K^s}^s\}$. For example, the *restaurant* domain has a slot named *price range*, and the possible values are `cheap`, `moderate`, and `expensive`. Some slots do not have pre-defined values, that is, V^s is missing in the domain ontology. For example, the *taxi* domain has a slot named *leave time*, but it is a poor choice to enumerate all the possible leave times the user may request as the size of V^s will be very large. Meanwhile, the domain ontology can also change over time. Formally, we represent a dialogue X as $X = \{U_1^a, U_1^u, U_2^a, U_2^u, ..., U_T^a, U_T^u\}$, where U_t^a is the agent utterance in turn t and U_t^u is the user utterance in turn t. Each turn t is associated with a dialogue state y_t. A dialogue state y_t is a set of (domain, slot, value) tuples. Each tuple represents that, up to the current turn t, a slot $s \in S^d$ of domain $d \in D$, which takes the value $v \in V^s$ has been provided by the user. Accordingly, y_t's are targets that the model needs to predict.

3 Multi-domain Dialogue State Tracking via Question Answering (DSTQA)

We model multi-domain DST as a question answering problem and use machine reading methods to provide answers. To predict the dialogue state at turn t, the model observes the context C_t, which is the concatenation of $\{U_1^a, U_1^u, ..., U_t^a, U_t^u\}$. The context is read by the model to answer the questions defined as follows. First, for each domain $d \in D$ and each slot $s \in S^d$ where there exists a pre-defined value set V^s, we construct a question $Q_{d,s} = \{d, s, V^s, \texttt{not mentioned}, \texttt{don't care}\}$. That is, a question is a set of words or phrases which includes a domain name, a slot name, a list of all possible values, and two special values `not mentioned` and `don't care`. One example of the constructed question for *restaurant* domain and *price range* slot is

$Q_{d,s} = \{$*restaurant*, *price range*, `cheap`, `moderate`, `expensive`, `not mentioned`, `don't care`$\}$. The constructed question represents the following natural language question: *"In the dialogue up to turn t, did the user mention the 'price range' of the 'restaurant' he/she is looking for? If so, which of the following option is correct: A)* `cheap`*, B)* `moderate`*, C)* `expensive`*, D)* `don't care`*."* As we can see from the above example, instead of only using domains and slots to construct questions (corresponding to natural language questions *what is the value of this slot?*), we also add candidate values V^s into $Q_{d,s}$, this is because values can be viewed as descriptions or complimentary information to domains and slots. For example, `cheap`, `moderate` and `expensive` explains what *price range* is. In this way, the constructed question $Q_{d,s}$ contains rich information about the domains and slots to predict, and easy to generalize to new values.

In the case that V^s is not available, the question is just the domain and slot names along with the special values, that is, $Q_{d,s} = \{d, s,$ `not mentioned`, `don't care`$\}$. For example, the constructed question for *train* domain and *leave time* slot is $Q_{d,s} = \{$*train*, *leave time*, `not mentioned`, `don't care`$\}$, and represents the following natural language question: *"In the dialogue up to turn t, did the user mention the 'leave time' of the 'train' he/she is looking for? If so, what is the 'leave time' the user preferred?"* The most important concept to note here is that the proposed DSTQA model can be easily extended to new domains, slots, and values. Tracking new domains and slots is simply constructing new queries, and tracking new values is simply extending the constructed question of an existing slot.

Although we formulate multi-domain dialogue state tracking as a question answering problem, we want to emphasize that there are some fundamental differences between these two settings. In a standard question answering problem, question understanding is a major challenge and the questions are highly dependent on the context where questions are often of many different forms (Rajpurkar et al., 2018). Meanwhile, in our formulation, the question forms are limited to two, every turn results in asking a restricted set of question types, and thus question understanding is straightforward. Conversely, our formulation has its own complicating characteristics including: (1) questions in consecutive turns tend to have the same answers, (2) an answer is either a span of the context or a value from a value set, and (3) the questions we constructed have some underlying connections defined by a dynamically-evolving knowledge graph (described in Section 4), which can help improve model performance. In any case, modeling multi-domain DST with this approach allows us to easily transfer knowledge to new domains, slots, and values simply by constructing new questions. Accordingly, many existing reading comprehension algorithms (Seo et al., 2017; Yu et al., 2018; Devlin et al., 2019; Clark & Gardner, 2018) can be directly applied here. In this paper, we propose a bidirectional attention flow (Seo et al., 2017) based model for multi-domain DST.

3.1 Model Overview

Figure 2 summarizes the DSTQA architecture, where notable subcomponents are detailed below. **1. Word Embedding Layer**: For each word in context C_t, similar to Seo et al. (2017), we apply a character embedding layer based on convolutional neural network to get a D^{Char} dimensional character-level embedding. We then adopt ELMo (Peters et al., 2018), a deep contextualized word representations, to get a D^{ELMo} dimensional word-level embedding. Other contextualized word embeddings such as BERT (Devlin et al., 2019) can also be applied here but is orthogonal to DSTQA and is left for future work. The final word embedding of context C_t is the concatenation of the character-level embedding and the ELMo embedding, and is denoted by $W^c \in \mathbb{R}^{L_c \times D^w}$, where L_c is the number of words in context C_t and $D^w = D^{\text{ELMo}} + D^{\text{Char}}$. Similarly, For a question $Q_{d,s}$, we treat each element in $Q_{d,s}$ (either a domain name, a slot name, or a value from the value set) as a sentence and compute its word embedding. We then take the mean of the word embeddings in each element as the embedding of that element. Then the question embedding is represented by a set $\{w^d \in \mathbb{R}^{D^w}, w^s \in \mathbb{R}^{D^w}, W^{\bar{v}} \in \mathbb{R}^{L_{\bar{v}} \times D^w}\}$, where w^d, w^s and $W^{\bar{v}}$ are domain, slot and value embeddings, respectively, and $L_{\bar{v}}$ is the number of values in V^s plus `not mentioned` and `don't care`. To represent the question embedding as one single matrix, we define $W^q \in \mathbb{R}^{L_{\bar{v}} \times D^w}$, where each row of W^q is calculauted by $W^q_{j,:} = w^d + w^s + W^{\bar{v}}_{j,:}$.

2. Context Encoding Layer: We apply a bidirectional GRU to encode the context C_t. Denoting the i-th word in the context C_t by w_i, then the input to the bidirectional GRU at time step i is the concatenation of the following three vectors: 1) w_i's word embeddings, $W^c_{i,:}$, 2) the corresponding role embedding, and 3) exact match features. There are two role embeddings: the agent role embed-

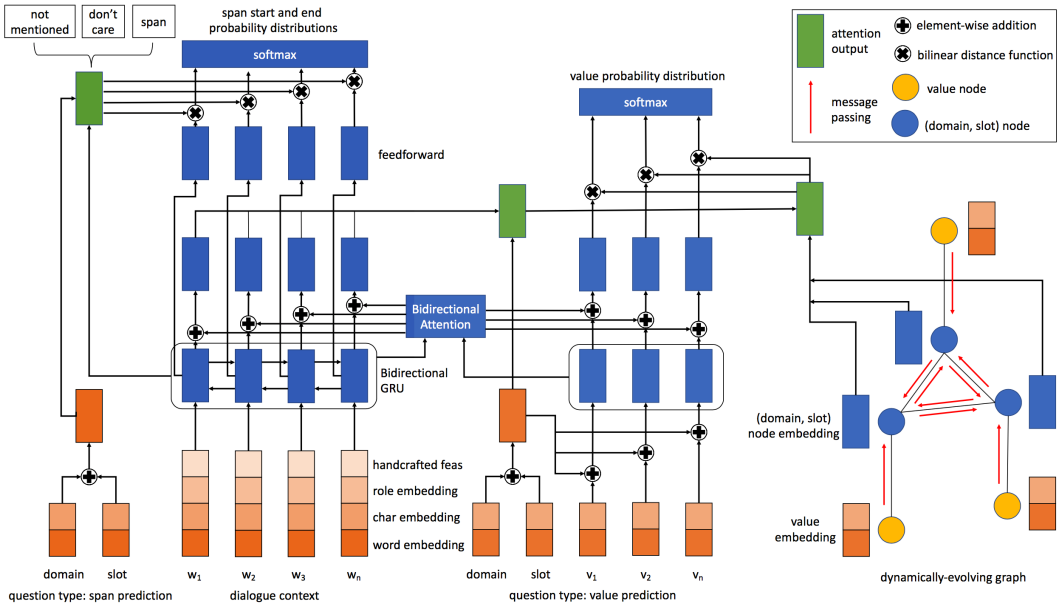

Figure 2: DSTQA model architecture. When the question type is value prediction, a bidirectional attention layer is applied to the dialogue context and the question, and a graph embedding is injected to the output of the bidirectional attention layer. When the question type is span prediction, the question is used to attend over the dialogue context to predict span start and end positions.

ding $e_a \in \mathbb{R}^r$ and the user role embedding $e_u \in \mathbb{R}^r$ where both are trainable. Exact match features are binary indicator features where for each (domain, slot) pair, we search for occurrences of its values in the context in original and lemmatized forms. Then for each (domain, slot) pair, we use two binary features to indicate whether w_i belongs to an occurrence in either form. The final output of this layer is a matrix $E^c \in \mathbb{R}^{L_c \times D^{\text{biGRU}}}$, where L_c is the number of words in the context C_t and D^{biGRU} is the dimension of bidirectional GRU's hidden states (includes both forward and backward hidden states). In our experiments, we set D^{biGRU} equals to D^w.

3. Question-Context Bidirectional Attention Layer: Inspired by Seo et al. (2017), we apply a bidirectional attention layer which computes attention in two directions: from context C_t to question $Q_{d,s}$, and from question $Q_{d,s}$ to context C_t. To do so, we first define an attention function $\mathbb{R}^{m*n} \times \mathbb{R}^n \to \mathbb{R}^m$ that will be used frequently in the following sections. The inputs to the function are a key matrix $K \in \mathbb{R}^{m*n}$ and a query vector $q \in \mathbb{R}^n$. The function calculates the attention score of q over each row of K. Let $O \in \mathbb{R}^{m*n}$ be a matrix which is q repeated by m times, that is, $O_{:,j} = q$ for all j. Then, the attention function is defined as:

$$\text{Att}_\beta(K, q) = \text{Softmax}([K; O^\top; K \odot O^\top] \cdot \beta)$$

Where $\beta \in \mathbb{R}^{3n}$ are learned model parameters, $\odot$ is the element-wise multiplication operator, and $[;]$ is matrix row concatenation operator. We use subscript of β, β_i, to indicate different instantiations of the attention function.

The attention score of a context word w_i to values in $Q_{d,s}$ is given by $\alpha_i^v = \text{Att}_{\beta_1}(W^q, E^c_{i,:}) \in \mathbb{R}^{L_{\bar{v}}}$, and the attention score of a value v_j to context words in C_t is given by $\alpha_j^w = \text{Att}_{\beta_1}(E^c, W^q_{j,:}) \in \mathbb{R}^{L_c}$. β_1 is shared between these two attention functions. Then, the question-dependent embedding of context word w_i is $B_i^{QD} = W^{q\top} \cdot \alpha_i^v$ and can be viewed as the representation of w_i in the vector space defined by the question $Q_{d,s}$. Similarly, the context-dependent embedding for value v_j is $B_j^{CD} = E^{c\top} \cdot \alpha_j^w$ and can be viewed as the representation of v_j in the vector space defined by the context C_t. The final context embedding is $B^c = E^c + B^{QD} \in \mathbb{R}^{L_c \times D^w}$ and the final question embedding is $B^q = B^{CD} + W^q \in \mathbb{R}^{L_{\bar{v}} \times D^w}$.

4. Value Prediction Layer: When V^s exists in $Q_{d,s}$, we calculate a score for each value in $Q_{d,s}$, and select the one with the highest score as the answer. First, we define a bilinear function $\mathbb{R}^{m*n} \times \mathbb{R}^n \to$

$\mathbb{R}^m$. It takes a matrix $X \in \mathbb{R}^{m*n}$ and a vector $y \in \mathbb{R}^n$, returning a vector of length m,

$$\text{BiLinear}_\Phi(X, y) = X^\top \Phi y$$

where $\Phi \in \mathbb{R}^{n*n}$ are learned model parameters. Again, we use subscript of Φ, Φ_i, to indicate different instantiations of the function.

We summarize context B^c into a single vector with respect to the domain and slot and then apply a bilinear function to calculate the score of each value. More specifically, We calculate the score of each value v at turn t by

$$p_t^v = \text{Softmax}\left(\text{BiLinear}_{\Phi_1}\left(B^q, {B^c}^\top \cdot \alpha^b\right)\right) \tag{1}$$

where $\alpha^b = \text{Att}_{\beta_2}(B^c, w^d + w^s) \in \mathbb{R}^{L_c}$ is the attention score over B^c, and $p_t^v \in \mathbb{R}^{L_{\bar{v}}}$. We calculate the cross entropy loss of the predicted scores by $\text{Loss}_v = \sum_t \sum_{d \in D, s \in \hat{S}^d} \text{CrossEntropy}\left(p_t^v, y_t^v\right)$ where $y_t^v \in \mathbb{R}^{L_{\bar{v}}}$ is the label, which is the one-hot encoding of the true value of domain d and slot s, and $\hat{S}^d$ is the set of slots in domain d that has pre-defined V^s.

5. Span Prediction Layer: When the value set V^s is unknown or too large to enumerate, such as *pick up time* in *taxi* domain, we predict the answer to a question $Q_{d,s}$ as either a span in the context or two special types: `not mentioned` and `don't care`. The span prediction layer has two components. The first component predicts the answer type of $Q_{d,s}$. The type of the answer is either `not mentioned`, `don't care` or `span`, and is calculated by $p_t^{st} = \text{Softmax}(\Theta_1 \cdot (w^d + w^s + {E^c}^\top \cdot \alpha^e))$ where $\alpha^e = \text{Att}_{\beta_3}(E^c, w^d + w^s) \in \mathbb{R}^{L_c}$, $\Theta_1 \in \mathbb{R}^{3*D^w}$ is a model parameter to learn, and $p_t^{st} \in \mathbb{R}^3$. The loss of span type prediction is $\text{Loss}_{st} = \sum_t \sum_{d \in D, s \in \bar{S}^d} \text{CrossEntropy}\left(p_t^{st}, y_t^{st}\right)$ where $y_t^{st} \in \mathbb{R}^3$ is the one-hot encoding of the true span type label, and $\bar{S}^d$ is the set of slots in domain d that has no pre-defined V^s. The second component predicts a span in the context corresponding to the answer of $Q_{d,s}$. To get the probability distribution of a span's start index, we apply a bilinear function between contexts and (domain, slot) pairs. More specifically, $p_t^{ss} = \text{Softmax}\left(\text{BiLinear}_{\Phi_2}\left(\text{Relu}\left(E^c \cdot \Theta_2\right), \left(w^d + w^s + {E^c}^\top \cdot \alpha^e\right)\right)\right)$ where $\Theta_2 \in \mathbb{R}^{D^w * D^w}$ and $p_t^{ss} \in \mathbb{R}^{L_c}$. The Bilinear function's first argument is a non-linear transformation of the context embedding, and its second argument is a context-dependent (domain, slot) pair embedding. Similarly, the probability distribution of a span's end index is $p_t^{se} = \text{Softmax}\left(\text{BiLinear}_{\Phi_3}\left(\text{Relu}\left(E^c \cdot \Theta_2 \cdot \Theta_3\right), \left(w^d + w^s + {E^c}^\top \cdot \alpha^e\right)\right)\right)$ where $\Theta_3 \in \mathbb{R}^{D^w * D^w}$ and $p_t^{se} \in \mathbb{R}^{L_c}$. The prediction loss is $\text{Loss}_{span} = \sum_t \sum_{d \in D, s \in \bar{S}^d} \text{CrossEntropy}(p_t^{ss}, y_t^{ss}) + \text{CrossEntropy}(p_t^{se}, y_t^{se})$ where $y_t^{ss}, y_t^{se} \in \mathbb{R}^{L_c}$ is one-hot encodings of true start and end indices, respectively. The score of a span is the multiplication of probabilities of its start and end index. The final loss function is: $\text{Loss} = \text{Loss}_v + \text{Loss}_{st} + \text{Loss}_{span}$. Most publicly available DST datasets do not have span start and end labels. In Appendix B we show how we construct these labels.

4 Dynamic Knowledge Graph for Multi-domain dialogue State Tracking

In our problem formulation, at each turn, our proposed algorithm asks a set of questions, one for each (domain, slot) pair. In fact, the (domain, slot) pairs are not independent. For example, if a user requested a train for 3 people, then the number of people for hotel reservation may also be 3. If a user booked a restaurant, then the destination of the taxi is likely to be that restaurant. Specifically, we observe four types of relationships between (domain, slot) pairs in MultiWOZ 2.0/2.1 dataset:

1. (s, r_v, s'): a slot $s \in S^d$ and another slot $s' \in S^{d'}$ have the same set of possible values. That is, V^s equals to $V^{s'}$. For example, in MultiWOZ 2.0/2.1 dataset, domain-slot pairs (*restaurant*, *book day*) and (*hotel*, *book day*) have this relationship.
2. (s, r_s, s'): the value set of a slot $s \in S^d$ is a subset of the value set of $s' \in S^{d'}$. For example, in MultiWOZ 2.0/2.1 dataset, value sets of (*restaurant*, *name*), (*hotel*, *name*), (*train*, *station*) and (*attraction*, *name*) are subsets of the value set of (*taxi*, *destination*).
3. (s, r_c, s'): the informed value $v \in V^s$ of slot s is correlated with the informed value $v \in V^{s'}$ of slot s' even though V^s and $V^{s'}$ do not overlap. For example, in MultiWOZ 2.0/2.1 dataset, the price range of a reserved restaurant is correlated with the star of the booked hotel. This relationship is not explicitly given in the ontology.

4. (s, r_i, v): the user has informed value $v \in V^s$ of slot $s \in S^d$.

In this section, we propose using a dynamic knowledge graph to further improve model performance by exploiting this information. We represent (domain, slot) pairs and values as nodes in a graph linked by the relationship defined above, and then propagate information between them. The graph is *dynamically evolving*, since the fourth relationship above, r_i, depends on the dialogue context.

4.1 Graph Definition

The right-hand side of Figure 2 is an example of the graph we defined based on the ontology. There are two types of nodes $\{M, N\}$ in the graph. One is a (domain, slot) pair node representing a (domain, slot) pair in the ontology and another is a value node representing a value from a value set. For a domain $d \in D$ and a slot $s \in S^d$, we denote the corresponding node by $M_{d,s}$, and for a value $v \in V^s$, we denote the corresponding node by N_v. There are also two types of edges. One type is the links between M and N. At each turn t, if the answer to question $Q_{d,s}$ is $v \in V^s$, then N_v is added to the graph and linked to $M_{d,s}$. By default, $M_{d,s}$ is linked to a special `not mentioned` node. The other type of edges is links between nodes in M. Ideally we want to link nodes in M based on the first three relationships described above. However, while r_v and r_s are known given the ontology, r_c is unknown and cannot be inferred just based on the ontology. As a result, we connect every node in M (i.e. the (domain, slot) pair nodes) with each other, and let the model to learn their relationships with an attention mechanism, which will be described shortly.

4.2 Attention Over the Graph

We use an attention mechanism to calculate the importance of a node's neighbors to that node, and then aggregate node embeddings based on attention scores. Veličković et al. (2018) describes a graph attention network, which performs self-attention over nodes. In contrast with their work, we use dialogue contexts to attend over nodes.

Our attention mechanism has two steps. The first step is to propagate the embedding of N_v to its linked $M_{d,s}$, so that the embedding of $M_{d,s}$ depends on the value prediction from previous turns. We propagate N_v's embedding by $g_{d,s} = \eta(w^d + w^s) + (1-\eta)\sigma(\Theta_4 \cdot W^{\bar{v}}_{v,:})$ where $g_{d,s} \in \mathbb{R}^{D^w}$ is the new embedding of $M_{d,s}$, $\eta \in [0, 1]$ is a hyper-parameter, and $\Theta_4 \in \mathbb{R}^{D^w \times D^w}$ is a model parameter to learn. $g_{d,s}$ essentially carries the following information: in previous turns, the user has mentioned value v of a slot s from a domain d. In practice, we find out that simply adding w^d, w^s and $W^{\bar{v}}$ yields the best result. That is $g_{d,s} = w^d + w^s + W^{\bar{v}}_{v,:}$. The second step is to propagate information between nodes in M. For each domain d and slot s, $B^{c\top} \cdot \alpha^b$ in Equation (1) is the summarized context embedding with respect to d and s. We use this vector to attend over all nodes in M, and the attention score is $\alpha^g = \text{Att}_{\beta_4}(G, B^{c\top} \cdot \alpha^b)$, where $G \in \mathbb{R}^{|M| * D^w}$ is a matrix stacked by $g^\top_{d,s}$. The attention scores can be interpreted as the learned relationships between the current (domain, slot) node and all other (domain, slot) nodes. Using context embeddings to attend over the graph allows the model to assign attention score of each node based on dialogue contexts. Finally, The graph embedding is $z_{d,s} = G \cdot \alpha^g$. We inject $z_{d,s}$ to Equation (1) with a gating mechanism:

$$p^v_t = \text{Softmax}\left(\text{BiLinear}_{\Phi_1}\left(B^q, (1-\gamma)B^{c\top} \cdot \alpha^b + \gamma z_{d,s}\right)\right) \tag{2}$$

where $\gamma = \sigma(B^{c\top} \cdot \alpha^b + z_{d,s})$ is the gate and controls how much graph information should flow to the context embedding given the dialogue context. Some utterances such as *"book a taxi to Cambridge station"* do not need information in the graph, while some utterances such as *"book a taxi from the hotel to the restaurant"* needs information from other domains. γ dynamically controls in what degree the graph embedding is used, and graph parameters are trained together with all other parameters.

5 Experiments

We evaluate our model on three publicly available datasets: (non-multi-domain) WOZ 2.0 (Mrkšić et al., 2017), MultiWOZ 2.0 (Budzianowski et al., 2018), and MultiWOZ 2.1 (Eric et al., 2019). Due to limited space, please refer to Appendix C.2 for results on (non-multi-domain) WOZ 2.0

dataset. MultiWOZ 2.0 dataset is collected from a Wizard of Oz style experiment and has 7 domains: *restaurant*, *hotel*, *train*, *attraction*, *taxi*, *hospital*, and *police*. Similar to Wu et al. (2019), we ignore the *hospital* and *police* domains because they only appear in training set. There are 30 (domain, slot) pairs and a total of 10438 task-oriented dialogues. A dialogue may span across multiple domains. For example, during the conversation, a user may book a restaurant first, and then book a taxi to that restaurant. For both datasets, we use the train/test splits provided by the dataset. The domain ontology of the datasets is described in Appendix C.3. MultiWOZ 2.1 contains the same dialogues and ontology as MultiWOZ 2.0, but fixes some annotation errors in MultiWOZ 2.0.

Two common metrics to evaluate dialogue state tracking performance are ***Joint*** accruacy and ***Slot*** accuracy. Joint accuracy is the accuracy of dialogue states. A dialogue state is correctly predicted only if all the values of (domain, slot) pairs are correctly predicted. Slot accuracy is the accuracy of (domain, slot, value) tuples. A tuple is correctly predicted only if the value of the (domain, slot) pair is correctly predicted. In most literature, joint accuracy is considered as a more challenging and more important metric. Implementation details are described in Appendix B.

5.1 Results on MultiWoz 2.0 and MultiWOZ 2.1 dataset.

We first evaluate our model on MultiWOZ 2.0 dataset as shown in Table 1. We compare with five published baselines. TRADE (Wu et al., 2019) is the current published state-of-the-art model. It utilizes an encoder-decoder architecture that takes dialogue contexts as source sentences, and takes state annotations as target sentences. SUMBT (Lee et al., 2019) finetunes a pre-trained BERT model (Devlin et al., 2019) to learn slot and utterance representations. Neural Reading (Gao et al., 2019) learns a question embedding for each slot, and predicts the span of each slot value. GCE (Nouri & Hosseini-Asl, 2018) is a model improved over GLAD (Zhong et al., 2018) by using a slot-conditioned global module. Details about baselines and related works are in Appendix A.

	Joint	Slot
GLAD	35.57	95.44
GCE	36.27	98.42
Neural Reading	42.12	-
SUMBT	46.65	96.44
TRADE	48.62	96.92
DSTQA w/span	**51.36**	97.22
-graph	50.89	97.17
-gating	50.38	97.14
-bi att +avg	49.74	97.11
-bi att	49.51	97.07
-ELMo +GloVe	49.52	96.96
DSTQA w/o span	**51.44**	97.24
-ELMo +GloVe	50.81	97.19

Table 1: Results on MultiWOZ 2.0

For our model, we report results under two settings. In the DSTQA w/span setting, we do span prediction for the five time related slots as mentioned in Appendix B. This is the most realistic setting as enumerating all possible time values is not practical in a production environment. In the DSTQA w/o span setting, we do value prediction for all slots, including the five time related slots. To do this, we collect all time values appeared in the training data to create a value list for time related slots as is done in baseline models. It works in these two datasets because there are only 173 time values in the training data, and only 14 out-of-vocabulary time values in the test data. Note that in all our baselines, values appeared in the training data are either added to the vocabulary or added to the domain ontology, so DSTQA w/o span is still a fair comparison with the baseline methods. Our model outperforms all models. DSTQA w/span has a 5.64% relative improvement and a 2.74% absolute improvement over TRADE. We also show the performance on each single domain in Appendix C.4. DSTQA w/o span has a 5.80% relative improvement and a 2.82% absolute improvement over TRADE. We can see that DSTQA w/o span performs better than DSTQA w/span, this is mainly because we introduce noises when constructing the span labels, meanwhile, span prediction cannot take the benefit of the bidirectional attention mechanism. However, DSTQA w/o span cannot handle out-of-vocabulary values, but can generalize to new values only by expanding the value sets, moreover, the performance of DSTQA w/o span may decrease when the size of value sets increases. Table 2 shows the results on MultiWOZ 2.1 dataset. Compared with TRADE, DSTQA w/span has a 8.93% relative improvement and a 4.07% absolute improvement. DSTQA w/o span has a 12.21% relative improvement and a 5.57% absolute improvement. More baselines can be found at the leaderboard.[1] Our model outperforms all models on the leaderboard at the time of submission of this paper.

[1] http://dialogue.mi.eng.cam.ac.uk/index.php/corpus/

Ablation Study: Table 1 also shows the results of ablation study of DSTQA w/span on MultiWOZ 2.0 dataset. The first experiment completely removes the graph component, and the joint accuracy drops 0.47%. The second experiment keeps the graph component but removes the gating mechanism, which is equivalent to setting γ in Equation (2) to 0.5, and the joint accuracy drops 0.98%, demonstrating that the gating mechanism is important when injecting graph embeddings and simply adding the graph embeddings to context embeddings can negatively impact the performance. In the third experiment, we replace B_i^{QD} with the mean of query word embeddings and replace B_j^{CD} with the mean of context word embeddings. This is equivalent to setting the bi-directional attention scores uniformly. The joint accuracy significantly drops 1.62%. The fourth experiment completely removes the bi-directional attention layer, and the joint accuracy drops 1.85%. Both experiments show that bidirectional attention layer has a notably positive impact on model performance. The fifth experiment substitute ELMo embeddings with GloVe embeddings to demonstrate the benefit of using contextual word embeddings. We plan to try other state-of-the-art contextual word embeddings such as BERT (Devlin et al., 2019) in the future. We further show the model performance on different context lengths in Appendix C.5.

	Joint	Slot
TRADE	45.60	-
DSTQA w/span	**49.67**	97.10
-graph	49.48	97.05
-ELMo +GloVe	48.15	96.98
DSTQA w/o span	**51.17**	97.21
-ELMo +GloVe	50.03	97.12

Table 2: Results on MultiWOZ 2.1

5.2 Error Analysis

Figure 3 shows the different types of model prediction errors on MultiWOZ 2.1 dataset made by DSTQA w/span as analyzed by the authors. Appendix C.7 explains the meaning of each error type and also list examples for each error type. At first glance, annotation errors and annotation disagreements account for 56% of total prediction errors, and are all due to noise in the dataset and thus unavoidable. *Annotation errors* are the most frequent errors and account for 28% of total prediction errors. Annotation errors means that the model predictions are incorrect only because the corresponding ground truth labels in the dataset are wrong. Usually this happens when the annotators neglect the value informed by the user. *Annotator disagreement on user confirmation* accounts for 28% ($15\% + 13\%$) of total errors. This type of errors comes from the disagreement between annotators when generating ground truth labels. All these errors are due to the noise in the dataset and unavoidable, which also explains why the task on MultiWOZ 2.1 dataset is challenging and the state-of-the-art joint accuracy is less than 50%.

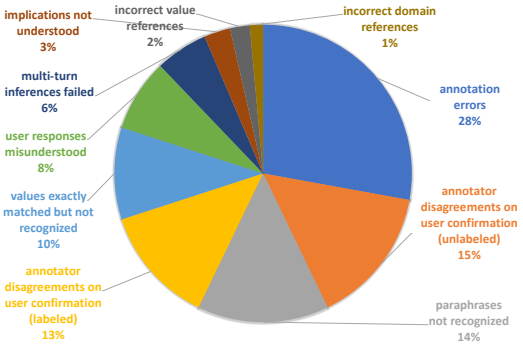

Figure 3: Error Types on MultiWOZ 2.1 dataset.

Values exactly matched but not recognized (10%) and *paraphrases not recognized* (14%) mean that the user mentions a value or a paraphrase of a value, but the model fails to recognize it. *Multi-turn inferences failed* (6%) means that the model fails to refer to previous utterances when making prediction. *User responses not understood* (8%) and *implications not understood* (3%) mean that the model does not understand what the user says and fails to predict based on user responses. Finally, *incorrect value references* (2%) means that there are multiple values of a slot in the context and the model refers to an incorrect one, and *incorrect domain references* (1%) means that the predicted slot and value should belong to another domain. All these errors indicate insufficient understanding of agent and user utterances. A more powerful language model and a coreference resolution modules may help mitigate these problems. Please refer to Appendix C.7 for examples.

6 Conclusion

In this paper, we model multi-domain DST as question answering with a dynamically-evolving knowledge graph. Such formulation enables the model to generalize to new domains, slots and values by simply constructing new questions. Our model achieves state-of-the-art results on MultiWOZ 2.0 and MultiWOZ 2.1 dataset with a 5.80% and a 12.21% relative improvement, respectively.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5016–5026, 2018.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879, 2017.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 845–855, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019.

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

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

Matthew Henderson, Blaise Thomson, and Steve Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pp. 360–365, 2014.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (3rd ed. draft)*. 2019.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 1777–1788, 2017.

Elnaz Nouri and Ehsan Hosseini-Asl. Toward scalable neural dialogue state tracking model. In *2nd Conversational AI workshop, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Julien Perez and Fei Liu. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 305–314, 2017.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 2227–2237, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 784–789, 2018.

Osman Ramadan, Paweł Budzianowski, and Milica Gasic. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 432–437, 2018.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 561–568. IEEE, 2017.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*, 2019.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. Towards universal dialogue state tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2780–2786, 2018.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *International Conference on Learning Representations*, 2017.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4231–4242, 2018.

Blaise Thomson and Steve Young. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

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

Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pp. 423–432, 2013.

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

Puyang Xu and Qi Hu. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1448–1457, 2018.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *International Conference on Learning Representations*, 2018.

Yuyu Zhang, Hanjun Dai, Kamil Toraman, and Le Song. Kgˆ 2: Learning to reason science exam questions with contextual knowledge graph embeddings. *arXiv preprint arXiv:1805.12393*, 2018.

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

A Related Works

Our work is most closely related to previous works in dialogue state tracking and question answering. Early models of dialogue state tracking (Thomson & Young, 2010; Wang & Lemon, 2013; Henderson et al., 2014) rely on handcrafted features to extract utterance semantics, and then use these features to predict dialogue states. Recently Mrkšić et al. (2017) propose to use convolutional neural network to learn utterance n-gram representation, and achieve better performance than handcrafted features-based model. However, their model maintains a separate set of parameters for each slot and does not scale well. Models that handles scalable multi-domain DST have then been proposed (Ramadan et al., 2018; Rastogi et al., 2017). Zhong et al. (2018) and Nouri & Hosseini-Asl (2018) propose a global-local architecture. The global module is shared by all slots to transfer knowledge between them. Ren et al. (2018) propose to share all parameters between slots and fix the word embeddings during training, so that they can handle new slots and values during inference. However, These models do not scale when the sizes of value sets are large or infinite, because they have to evaluate every (domain, slot, tuple) during the training. Xu & Hu (2018) propose to use a pointer network with a Seq2Seq architecture to handle unseen slot values. Lee et al. (2019) encode slots and utterances with a pre-trained BERT model, and then use a slot utterance matching module, which is a multi-head attention layer, to compute the similarity between slot values and utterances. Rastogi et al. (2019) release a schema-guided DST dataset which contains natural language description of domains and slots. They also propose to use BERT to encode these natural language description as embeddings of domains and slots. Wu et al. (2019) propose to use an encoder-decoder architecture with a pointer network. The source sentences are dialogue contexts and the target sentences are annotated value labels. The model shares parameters across domains and does not require pre-defined domain ontology, so it can adapt to unseen domains, slots and values. Our work differs in that we formulate multi-domain DST as a question answering problem and use reading comprehension methods to provide answers. There have already been a few recent works focusing on using reading comprehension models for dialogue state tracking. For example, Perez & Liu (2017) formulate slot tracking as four different types of questions (Factoid, Yes/No, Indefinite knowledge, Counting and Lists/Sets), and use memory network to do reasoning and to predict answers. Gao et al. (2019) construct a question for each slot, which basically asks *what is the value of slot i*, then they predict the span of the value/answer in the dialogue history. Our model is different from these two models in question representation. We not only use domains and slots but also use lists of candidate values to construct questions. Values can be viewed as descriptions to domains and slots, so that the questions we formulate have richer information about domains and slots, and can better generalize to new domains, slots, and values. Moreover, our model can do both span and value prediction, depending on whether the corresponding value lists exists or not. Finally, our model uses a dynamically-involving knowledge graph to explicitly capture interactions between domains and slots.

In a reading comprehension (Rajpurkar et al., 2016) task, there is one or more context paragraphs and a set of questions. The task is to answer questions based on the context paragraphs. Usually, an answer is a text span in a context paragraph. Many reading comprehension models have been proposed (Seo et al., 2017; Yu et al., 2018; Devlin et al., 2019; Clark & Gardner, 2018; Chen et al., 2017). These models encode questions and contexts with multiple layers of attention-based blocks and predict answer spans based on the learned question and context embeddings. Some works also explore to further improve model performance by knowledge graph. For example Sun et al. (2018) propose to build a heterogeneous graph in which the nodes are knowledge base entities and context paragraphs, and nodes are linked by entity relationships and entity mentions in the contexts. Zhang et al. (2018) propose to use Open IE to extract relation triples from context paragraphs and build a contextual knowledge graph with respect to the question and context paragraphs. We would expect many of these technical innovations to apply given our QA-based formulation.

B Implementation Details

Existing dialogue state tracking datasets, such as MultiWOZ 2.0 and MultiWOZ 2.1, do not have annotated span labels but only have annotated value labels for slots. As a result, we preprocess MultiWOZ 2.0 and MultiWOZ 2.1 dataset to convert value labels to span labels: we take a value label in the annotation, and search for its last occurrence in the dialogue context, and use that occurrence as span start and end labels. There are 30 slots in MultiWOZ 2.0/2.1 dataset, and 5 of them are time related slots such as *restaurant book time* and *train arrive by*, and the values are 24-hour clock time

Domain Expansion	Train on 5% Data from Scratch		Train on 5% Data by Fine Tuning			Train on 10% Data From Scratch		Train on 10% Data by Fine Tuning		
	TRADE	DSTQA w/o graph	TRADE	DSTQA w/o graph	DSTQA w/ graph	TRADE	DSTQA w/o graph	TRADE	DSTQA w/o graph	DSTQA w/ graph
Restaurant	47.31	35.33	55.70	58.89	**58.95**	53.65	54.27	60.94	**64.51**	64.48
Hotel	31.93	33.08	37.45	48.94	**50.18**	41.29	49.69	41.42	52.59	**53.68**
Train	48.82	50.36	69.27	69.32	**70.35**	59.65	61.28	71.11	73.74	**74.50**
Attraction	52.19	51.58	57.55	**70.47**	70.10	58.46	61.77	63.12	**71.60**	71.28
Taxi	59.03	58.25	66.58	68.19	**70.90**	60.51	59.35	70.19	72.52	**74.19**

Table 3: Joint accuracy on domain expansion experiments. Models are either trained from scratch on the target domain, or trained from the 4 source domains and then fine-tuned on the target domain.

such as `08:15`. We do span prediction for these 5 slots and do value prediction for the rest of slots because it is not practical to enumerate all time values. We can also do span prediction for other slots such as *restaurant name* and *hotel name* with the benefit of handling out-of-vocabulary values, but we leave these experiments as future work. WOZ 2.0 dataset only has one domain and 3 slots, and we do value prediction for all these slots without graph embeddings.

We implement our model using AllenNLP (Gardner et al., 2017) framework.[2] For experiments with ELMo embeddings, we use a pre-trained ELMo model[3] in which the output size is $D^{ELMo} = 512$. The dimension of character-level embeddings is $D^{Char} = 100$, making $D^w = 612$. ELMo embeddings are fixed during training. For experiments with GloVe embeddings, we use GloVe embeddings pre-trained on Common Crawl dataset.[4] The dimension of GloVe embeddings is 300, and the dimension of character-level embeddings is 100, such that $D^w = 400$. GloVe embeddings are trainable during training. The size of the role embedding is 128. The dropout rate is set to 0.5. We use Adam as the optimizer and the learning rate is set to 0.001. We also apply word dropout that randomly drop out words in dialogue context with probability 0.1.

When training DSTQA with the dynamic knowledge graph, in order to predict the dialogue state and calculate the loss at turn t, we use the model with current parameters to predict the dialogue state up until turn $t-1$, and dynamically construct a graph for turn t. We have also tried to do teacher forcing which constructs the graph with ground truth labels (or sample ground truth labels with an annealed probability), but we observe a negative impact on joint accuracy. On the other hand, target network (Mnih et al., 2015) may be useful here and will be investigated in the future. More specifically, we can have a copy of the model that update periodically, and use this model copy to predict dialogue state up until turn $t-1$ and construct the graph.

C Additional Results

C.1 Generalization to New Domains

Table 3 shows the model performance on new domains. We take one domain in MultiWOZ 2.0 as the target domain, and the remaining 4 domains as source domains. Models are trained either from scratch using only 5% or 10% sampled data from the target domain, or first trained on the 4 source domains and then fine-tuned on the target domain with sampled data. In general, a model that achieves higher accuracy by fine-tuning is more desirable, as it indicates that the model can quickly adapt to new domains given limited data from the new domain. In this experiment, we compare DSTQA w/span with TRADE. As shown in Table 3, DSTQA consistently outperforms TRADE when fine-tuning on 5% and 10% new domain data. With 5% new domain data, DSTQA fine-tuning has an average of 43.32% relative improvement over DSTQA training from scratch, while TRADE fine-tuning only has an average of 19.99% relative improvement over TRADE training from scratch. DSTQA w/ graph also demonstrates its benefit over DSTQA w/o graph, especially on the taxi domain. This is because the 'taxi' domain is usually mentioned at the latter part of the dialogue, and the destination and departure of the taxi are usually the restaurant, hotel, or attraction mentioned in the previous turns and are embedded in the graph.

C.2 Results on WOZ 2.0 dataset

We also evaluate our algorithm on WOZ 2.0 dataset (Mrkšić et al., 2017)

[2]Code will be released on Github

[3]https://allennlp.org/elmo

[4]https://nlp.stanford.edu/projects/glove/

WOZ 2.0 dataset has 1200 restaurant domain task-oriented dialogues. There are three slots: 'food', 'area', 'price range', and a total of 91 slot values. The dialogues are collected from a Wizard of Oz style experiment, in which the task is to find a restaurant that matches the slot values the user has specified. Each turn of a dialogue is annotated with a dialogue state, which indicates the slot values the user has informed. One example of the dialogue state is {*'food:Mexican', 'area':'east', price range:'moderate'*}.

Model	Joint Accuracy
NBT	84.4
GLAD	88.1
GCE	88.5
StateNet PSI	88.9
SUMBT	**91.00**
DSTQA	90.0

Table 4: Joint accuracy on WOZ 2.0 dataset.

Table 4 shows the results on WOZ 2.0 dataset. We compare with four published baselines. SUMBT (Lee et al., 2019) is the current state-of-the-art model on WOZ 2.0 dataset. It fine-tunes a pre-trained BERT model (Devlin et al., 2019) to learn slot and utterance representations. StateNet PSI (Ren et al., 2018) maps contextualized slot embeddings and value embeddings into the same vector space, and calculate the Euclidean distance between these two. It also learns a joint model of all slots, enabling parameter sharing between slots. GLAD (Zhong et al., 2018) proposes to use a global module to share parameters between slots and a local module to learn slot-specific features. Neural Beflief Tracker (Mrkšić et al., 2017) applies CNN to learn n-gram utterance representations. Unlike prior works that transfer knowledge between slots by sharing parameters, our model implicitly transfers knowledge by formulating each slot as a question and learning to answer all the questions. Our model has a 1.24% relative joint accuracy improvement over StateNet PSI. Although SUMBT achieves higher joint accuracy than DSTQA on WOZ 2.0 dataset, DSTQA achieves better performance than SUMBT on MultiWOZ 2.0 dataset, which is a more challenging dataset.

C.3 MultiWOZ 2.0/2.1 Ontology

The ontology of MultiWOZ 2.0 and MultiWOZ 2.1 datasets is shown in Table 5. There are 5 domains and 30 slots in total. (two other domains 'hospital' and 'police' are ignored as they only exists in training set.)

Domains	Restaurant	Hotel	Train	Attraction	Taxi
Slots	name area price range food book people book time book day	name area price range type parking stars internet book stay book day book people	destination departure day arrive by leave at book people	name area type	destination departure arrive by leave at

Table 5: Domain ontology in MultiWOZ 2.0 and MultiWOZ 2.1 dataset

C.4 Performance on Each Individual Domain

	Joint Accuray		Slot Accuracy	
	TRADE	DSTQA w/span	TRADE	DSTQA w/span
Restaurant	65.35	**68.68**	93.28	94.08
Hotel	55.52	**61.76**	92.66	93.72
Train	77.71	**79.75**	95.30	95.61
Attraction	71.64	**74.05**	88.97	90.53
Taxi	76.13	**78.22**	89.53	90.37

Table 6: Model performance on each of the 5 domains.

We show the performance of DSTQA w/span and TRADE on each single domain. We follow the same procedure as Wu et al. (2019) to construct training and test dataset for each domain: a dialogue

is excluded from a domain's training and test datasets if it does not mention any slots from that domain. During the training, slots from other domains are ignored. Table 6 shows the results. We can see that our model achieves better results on every domain, especially the hotel domain, which has a 11.24% relative improvement. Hotel is the hardest domain as it has the most slots (10 slots) and has the lowest joint accuracy among all domains.

C.5 Joint Accuracy v.s. Context Length

We further show the model performance on different context lengths. Context lengths means the number of previous turns included in the dialogue context. Note that our baseline algorithms either use all previous turns as contexts to predict belief states or accumulate turn-level states of all previous turns to generate belief states. The results are shown in Figure 4. We can see that DSTQA with graph outperforms DSTQA without graph. This is especially true when the context length is short. This is because when the context length is short, graph carries information over multiple turns which can be used for multi-turn inference. This is especially useful when we want a shorter context length to reduce computational cost. In this experiment, the DSTQA model we use is DSTQA w/span.

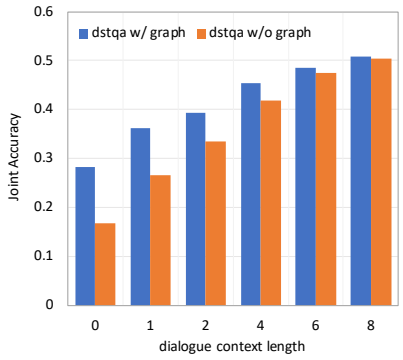

Figure 4: Joint acc. v.s. context length

C.6 Accuracy per Slot

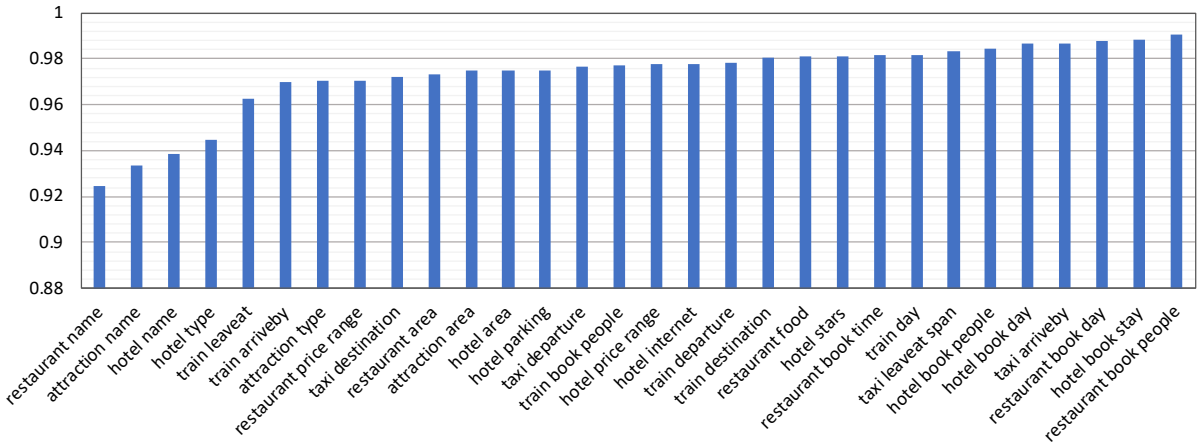

Figure 5: Accuracy of each slot per turn on MultiWOZ 2.0 dataset

The accuracy of each slot on MultiWOZ 2.0 and MultiWOZ 2.1 test set is shown in Figure 5 and Figure 6, respectively. Named related slots such as *restaurant name*, *attraction name*, *hotel name* has high error rate, because these slots have very large value set and high annotation errors.

C.7 Examples of Prediction Errors

This section describes prediciton errors made by DSTQA w/span. Incorrectly predicted (domain, slot, value) tuples are marked by underlines.

1. Annotation errors
Description: The groud truth label in the dataset is wrong. This can happen either 1) annotators neglect slots mentioned in the user utterance 2) annotators mistakenly choose the wrong label of a slot.
Examples:

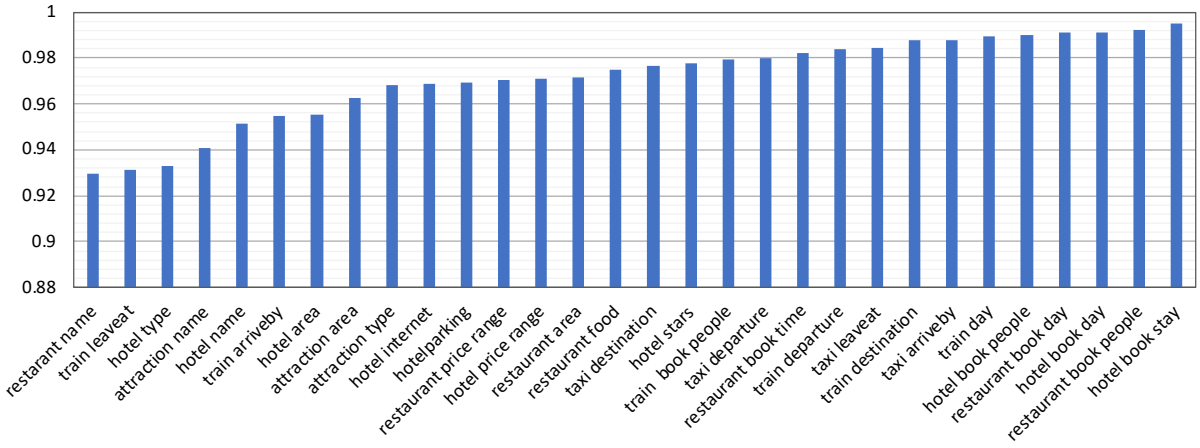

Figure 6: Accuracy of each slot per turn on MultiWOZ 2.1 dataset

User: I would like to find a *museum* in the *west* to go to. Agent: There are several museums in the west. I recommend the *Cafe Jello Gallery*. User: Can I have the address of the Cafe Jello museum? Agent: The Cafe Jello Gallery is at 13 Magdalene street. Is there anything else? User: Is there a *moderately* priced *British* restaurant *any where* in town?
Annotation: {(restaurant, food, British), (restaurant, price range, moderate), (restaurant, area, west)} **Prediction**: {(restaurant, food, Biritsh), (restaurant, price range, moderate), (restaurant, area, don't care)}

2. Annotator disagreement on user confirmation (labeled)
Description: This type of errors comes from the disagreement between annotators when generating ground truth labels. More specifically, in a dialogue, the agent sometimes proposes a suggestion (a value of a slot) to the user, followed by the user's positive confirmation. For example, the agent says 'I would recommend Little Seoul. Would you like to make a reservation?'. The user confirms with 'yes, please'. Since the user positively confirms the agent's suggestion, the (domain, slot, value) tuple mentioned by the agent, or, (restaurant, name, Little Seoul) tuple in this example, can be added into the belief state. However, based on our observation of the MultiWOZ 2.0 and MultiWOZ 2.1 dataset, the annotators are inconsistent, and only about half of the times these tuples are added to the belief states. An error of this type comes from the scenario that the tuple is added to the belief state by the annotator but not by the model (i.e. the model predicts 'None' for the corresponding (domain, slot) pair).
Examples:

User: I would like to find an *architectural attraction* to go to in the *centre*. Can you help me? Agent: Yes! I found 5 attractions matching your requirements . Would you like to know more? User: I would like to get the entrance fee and address of them please. Agent: *Great Saint Mary's Church* is 2 pounds and the other 4 are free. Would you like the address for Great Saint Mary's church? User: *Yes*, that would be great.
Annotation: {(attraction, type, architecture), (attraction, area, centre), (attraction, name, Great Saint Mary's Church)} **Prediction**: {(attraction, type, architecture), (attraction, area, centre)}

3. Annotator disagreement on user confirmation (unlabeled)
Description: Similar to the last type of errors, this one comes from the scenario that the tuple is added to the belief state by the model but not by the annotator.
Examples:

Agent: There are many great restaurants in the *centre*. Is there a certain type of food you would like to have?
User: What do you have that is *expensive*?
Agent: Yes, there are many. Would you be interested in trying British cuisine or African?
User: *British cuisine* is fine. Can you suggest a restaurant?
Agent: *Fitzbillies restaurant* meets all of your requirements. Would you like more info on them?
User: Book this for *6 people* on *Monday* at *12:30*.

Annotation: {(restaurant, area, centre), (restaurant, food, British), (restaurant, price range, expensive), (restaurant, time, 12:30), (restaurant, book people, 6), (restaurant, day, Monday)}
Prediction: {(restaurant, area, centre), (restaurant, food, British), (restaurant, price range, expensive), (restaurant, time, 12:30), (restaurant, book people, 6), (restaurant, day, Monday), (restaurant, name, Fitzbillies restaurant)}

4. Paraphrases not recognized
Description: The paraphrases of a value is not recognized by the model.
Example:

User: Can you help me find a place to go in the centre?
Agent: I can help you with that. Is there a certain kind of attraction that you would like to visit?
User: Surprise me. Give me the postcode as well.

Annotation: {(attraction, area, centre), (attraction, area, don't care)}
Prediction: {(attraction, area, centre)}

5. Value exactly matched but not recognized
Description: The value of a slot is mentioned and exactly matched in the user's utterance, but the model fails recognize and predict it.
Examples:

Agent: I am sorry, there is no restaurant serving specifically North American or American food in my database, is there another type of food you would consider?
User: How about *Modern European* food?
Agent: There are 3 Modern European restaurants. Two in the centre and one in the south. Do you have a preference?
User: I would prefer the one on the *centre*, could I have the phone number and postcode please?

Annotation: {(restaurant, food, Modern European), (restaurant, area, centre)}
Prediction: {(restaurant, food, Modern European)}

6. User responses misunderstood
Description: The model misunderstands the user's intention and fails to predict based on the user utterance.
Examples:

User: I could use some help finding a restaurant that is moderately priced.
Agent: We have many options that are *moderately priced*. Is there a specific area or type of cuisine you are looking for?
User: I do not care about the cuisine but I want it to be *in the west*.
Agent: We have *Prezzo*. It is an Italian restaurant located in the west. it is moderately priced. Would you like me to book it for you?
User: *That will not be necessary*. What is the postcode?
Agent: Prezzo's postcode is cb30ad.

Annotation: {(restaurant, price range, moderate), (restaurant, area, west)}
Prediction: {(restaurant, price range, moderate), (restaurant, area, west), (restaurant, name, Prezzo)}

7. Multi-turn inference failed
Description: In this scenario, it requires information from multiple turns to predict the value of a

slot, but the model fails to perform multi-turn inference.
Example:

User: Hello, may I have a list of museums in the west?
Agent: There are 7: Cafe Jello Gallery, Cambridge and County Folk Museum, ...
User: Please give me the entrance fee and postcode of County Folk Museum
Agent: The entrance fee is 3.50 pounds and the postcode is cb30aq. Would you like any other information?
User: I need a place to eat *near the museum*. I do not want to spend much so it should be *cheap*. what do you have?

Annotation: {(attraction, area, west), (attraction, type, museum), (attraction, name, Cambridge and County Folk Museum), (restaurant, price range, cheap), (restaurant, area, centre)}
Prediction: {(attraction, area, west), (attraction, type, museum), (attraction, name, Cambridge and County Folk Museum), (restaurant, price range, cheap)}

8. Implication not understood
Description: Implication expressed by the user is not understood by the model.
Examples:

User: I am trying to find a train leaving after *14:45* that's heading out *from London Liverpool street*. What do you have?
Agent: There are 45 trains that fit your criteria. Please clarify your destination, day of travel and the time you want to arrive by so that i can narrow it down.
User: I need a train *to Cambridge* on Tuesday.
Agent: I have 5 departures fitting your criteria on the :39 of the hour from 15:39 to 23:39. Would you like me to book any of these for you ?
User: Yes please do book the 15:39.

Annotation: {(train, leaveat, 14:45), (train, departure, London Liverpool street), (train, destination, Cambridge), (train, day, Tuesday), (train, book people, 1)}
Prediction: {(train, leaveat, 14:45), (train, departure, London Liverpool street), (train, destination, Cambridge), (train, day, Tuesday)}

9. Incorrect value reference
Description: There are multiple values of a slot in the context and the model refers to an incorrect one. This usually happens in time-related slots such as train departure time.
Examples:

User: I need to travel on *Saturday* from *Cambridge* to *London Kings Cross* and need to leave after *18:30*.
Agent: Train tr0427 leaves at 19:00 on Saturday and will get you there by 19:51. the cost is 18.88 pounds. Want me to book it?
User: Yes, please book the train for *1 person* and provide the reference number.

Annotation: {(train, departure, Cambridge), (train, destination, London King Cross), (train, day, Saturday), (train, book people, 1), (train, leaveat, 18:30)}
Prediction: {(train, departure, Cambridge), (train, destination, London King Cross), (train, day, Saturday), (train, book people, 1), (train, leaveat, 19:00)}

10. Incorrect domain reference
Description: The predicted slot and value should belong to another domain. This happens because many slots exists in multiple domains.
Example:

User: I am looking for information on Cambridge University Botanic Gardens.
Agent: They are on Bateman st., postal code cb21jf. They can be reached at 01223336265, the entrance fee is 4 pounds. Can I help with anything else?
User: Yes, can you help me find a restaurant?
Agent: The botanic gardens are in the centre . Would you like the restaurant to also be in the centre? do you have any type of cuisine in mind?
User: *never mind, i will worry about food later*. I am actually looking for a hotel with a *guesthouse* and *free parking* would be great as well.
Agent: There are 21 guesthouses with free parking, do you have a price or area preference?
User: *cheap* and *in the south* please .

Annotation: {(hotel, area, south), (hotel, parking, yes), (hotel, price range, cheap), (hotel, type, guesthouse)}
Prediction: {(hotel, area, south), (hotel, parking, yes), (hotel, price range, cheap), (hotel, type, guesthouse), (restaurant, price range, cheap)}