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: $\text{Loss} = \text{Loss}_v + \text{Loss}_{st} + \text{Loss}_{span}$. In most publicly available dialogue state tracking datasets, span start and end labels do not exist. In Section 5.1 we will 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.