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.