`care`. One example of the constructed question for *restaurant* domain and *price range* slot is $Q_{d,s} = \{\textit{restaurant}, \textit{price range}, \texttt{cheap}, \texttt{moderate}, \texttt{expensive}, \texttt{not mentioned}, \texttt{don't care}\}$. 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, $Q_{d,s} = \{d, s, \texttt{not mentioned}, \texttt{don't care}\}$. For example, the constructed question for *train* domain and *leave time* slot is $Q_{d,s} = \{\textit{train}, \textit{leave time}, \texttt{not mentioned}, \texttt{don't care}\}$, 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 $W^q_{j,:} = w^d + w^s + W^{\bar{v}}_{j,:}$.