# MA-DST: Multi-Attention-Based Scalable Dialog State Tracking

**Adarsh Kumar**
University of Wisconsin-Madison
adarsh@cs.wisc.edu

**Peter Ku**
Amazon Alexa AI

**Anuj Goyal**
Amazon Alexa AI

**Angeliki Metallinou**
Amazon Alexa AI

**Dilek Hakkani-Tur**
Amazon Alexa AI

## Abstract

Task oriented dialog agents provide a natural language interface for users to complete their goal. Dialog State Tracking (DST), which is often a core component of these systems, tracks the system's understanding of the user's goal throughout the conversation. To enable accurate multi-domain DST, the model needs to encode dependencies between past utterances and slot semantics and understand the dialog context, including long-range cross-domain references. We introduce a novel architecture for this task to encode the conversation history and slot semantics more robustly by using attention mechanisms at multiple granularities. In particular, we use cross-attention to model relationships between the context and slots at different semantic levels and self-attention to resolve cross-domain coreferences. In addition, our proposed architecture does not rely on knowing the domain ontologies beforehand and can also be used in a zero-shot setting for new domains or unseen slot values. Our model improves the joint goal accuracy by 5% (absolute) in the full-data setting and by up to 2% (absolute) in the zero-shot setting over the present state-of-the-art on the MultiWoZ 2.1 dataset.

## 1 Introduction

Task-oriented dialog systems provide users with a natural language interface to achieve a goal. Modern dialog systems support complex goals that may span multiple domains. For example, during the dialog the user may ask for a hotel reservation (hotel domain) and also a taxi ride to the hotel (taxi domain), as illustrated in the example of Figure 1. Dialog state tracking is one of the core components of task-oriented dialog systems. The dialog state can be thought as the system's belief of user's goal given the conversation history. For each user turn, the dialog state commonly includes the set of slot-value pairs, for all the slots which are mentioned by the user. An example is shown in Figure 1. Accurate DST is critical for task-oriented dialog as most dialog systems rely on such a state to predict the optimal next system action, such as a database query or a natural language generation (NLG) response.


Figure 1: Sample multi-domain dialog, spanning hotel and taxi domains, along with its dialog state

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Dialog state tracking requires understanding the semantics of the agent and user dialog so far, a challenging task since a dialog may span multiple domains and may include user or system references to slots happening earlier in the dialog. Data scarcity is an additional challenge, because dialog data collection is a costly and time consuming [15, 16]. As a result, it is critical to be able to train DST systems for new domains with zero or little data.

Previous work formulates DST as a classification task over all possible slot values for each slot, assuming all values are available in advance (e.g through a pre-defined ontology) [20, 10, 18]. However, to be most useful, DST systems should be able to track the values of even free-form slots such as "$hotel-name$", which typically contain out-of-vocabulary words. To overcome the limitations of ontology-based approaches candidate-set generation based approaches have been proposed [23]. TRADE [33] extends this idea further and propose a decoder-based approach that uses both generation and a pointing mechanism, taking a weighted sum of a distribution over the vocabulary and a distribution over the words in the conversation history. This enables the model to produce unseen slot values, and it achieves state-of-the art results on the MultiWOZ public benchmark [3, 8].

We extend this prior work by [33] and focus on improving the encoding of dialog context and slot semantics for DST to robustly capture important dependencies between slots and the conversation history as well as long-range coreferences in the conversation history. For this purpose, we propose a Multi-Attention DST (MA-DST) network. It contains multiple layers of cross-attention between the slot encodings and the conversation history to capture relationships at different levels of granularity, which are then followed by a self-attention layer to help resolve references to earlier slot mentions in the dialog. We show that the proposed MA-DST leads to an absolute improvement of over 5% in the joint goal accuracy over the current state-of-the art for the MultiWOZ 2.1 dataset in the full-data setting. We also show that MA-DST can be adapted to new domains with no training data in that new domain, achieving upto a 2% absolute joint goal accuracy gains on MultiWOZ 2.1 in the zero-shot setting.

## 2 Related Work

Dialog state tracking (DST) is a core dialog systems problem that is well studied in the literature, e.g. through the dialog state tracking challenges (DSTC) [12]. Earlier approaches for DST relied on Markov decision processes (MDPs) [17] and partially observable MDPs (POMDPs) [32, 25] for estimating the state updates. See [30] for a review of DST challenges and earlier related work.

Recent neural state tracking approaches achieve state-of-the-art performance on DST (See [9] for a review). Some of this work formulates the state tracking problem as a classification task over all possible slot-values per slot [20, 29, 18]. This assumes that an ontology containing all slot values per slot is available in advance. In practice, this is a limiting assumption, especially for free-form slots that may contain values not seen during training [34]. To address this limitation, [23] propose a candidate generation approach based on a bi-GRU network, that selects and scores slot values from the conversation history. [34] propose using a pointer network [27] for extracting slot values from the history. More recently, hybrid approaches which combine the candidate-set and slot-value generation approaches have appeared [11, 33].

Our work is most similar to TRADE [33], and extends it by proposing self [5] and cross-attention [2] mechanism for capturing slot and history correlations. Attention based archirectures like the Transformer [26] and architectures that extend it, like BERT [7] and RoBERTa [19], achieve the current state-of-the-arts for many NLP tasks. We are also inspired by the work in reading comprehension where cross attention is used to compute relations between a long passage and a query question [35, 4].

For benchmarking, DSTC challenges provide a popular experimentation framework and dialog data collected through human-machine interactions. Initially, they focused on single domain systems like bus routes [31], and restaurants [12]. Wizard-of-Oz (WOZ) is also a popular framework used to collect human-human dialogs that reflect the target human-machine behavior [29, 1]. Recently, the MultiWOZ 2.0 dataset, collected through WOZ for multiple domains, was introduced to address the lack of a large multi-domain DST benchmark [3]. [8] released an updated version, called MultiWOZ 2.1, which contains annotation corrections and new benchmark results using the current state-of-the-art approaches. Here, we use the MultiWOZ 2.1 dataset as our benchmark.

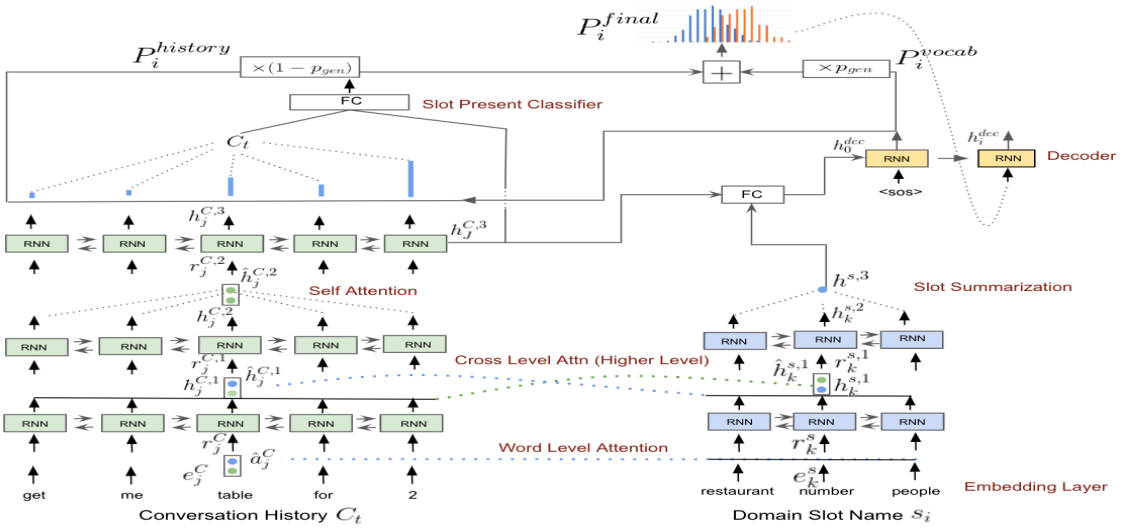


Figure 2: Model Architecture

# 3 Model Architecture

## 3.1 Problem Statement

Let's denote conversation history till turn $t$ as $C_t = \{U_1, A_1, U_2, A_2, ...U_t\}$, where $U_i$ and $A_i$ represents the user's utterance and agent's response at the $i^{th}$ turn. Let $S = \{s_1, s_2, ..., s_n\}$ denote the set of all $n$ possible slots across all domains. Let $DST_t = \{s_1 : v_1, s_2 : v_2, ..., s_n : v_n\}$ denote the dialog state at turn $t$, which contains all slots $s_i$ and their corresponding values $v_i$. Slots that are not mentioned in the dialog history take a "$none$" value. DST consists of predicting slot values for all slots $s_i$ at each turn $t$, given the conversation history $C_t$.

## 3.2 Model Architecture Overview

We build a neural model that encodes both the slot name $s_i$ and the conversation history so far $C_t$, and then decodes the slot value $v_i$, outputting words or special symbols for "$none$" and "$dontcare$" values. Our proposed model consists of an encoder $\text{Enc}_{slot}$ for the slot name, an encoder $\text{Enc}_{conv}$ for the conversation history, a decoder $\text{Dec}_{gen}$ that generates the slot value, and a three-class "slot gate" classifier $SG$ that predicts special symbols $\{none, dontcare, gen\}$, which will be described in detail later on. The model weights are shared between the slots, which makes the model more robust [33, 23] and scalable.

This architecture is similar to [33]. We propose modifications to the encoders in order to capture more fine grained dependencies between the slot name and the conversation history. Also, note the domain and slot names are concatenated into a single slot description, which we refer to as slot name for simplicity, and encoded via the slot encoder $\text{Enc}_{slot}$. Figure 2 illustrates the proposed architecture which we refer to as Multi-Attention DST (MA-DST).

## 3.3 Encoders

Our proposed slot $s_i$ and conversation history $C_t$ encoders use three stages of attention, specifically low-level cross-attention on the words, higher level cross-attention on the hidden state representations, and self-attention within the dialog history. Below we describe the encoders bottom-up.

### 3.3.1 Enriched Word Embedding Layer

For both $C_t$ and $s_i$, we first project each word into a low-dimensional space. We use a 300-dimensional GloVE embedding [21], and a 100-dimensional character embedding, both of which get fine-tuned during training. For the conversation history $C_t$, we also add a 5-dimensional POS tag embedding and a 5-dimensional NER

tag embeddings. We also use the turn index for each word as a feature and initialize it as a 5-dimensional embedding. We generated the POS tag and NER tag embeddings using the SpaCy library [13].

To capture the contextual meaning of words, we additionally use contextual ELMo embeddings [22]. We compute 1024-dimensional ELMo embeddings for both $C_t$ and $s_i$ by taking a weighted average of the different ELMo layers' outputs. Instead of fine-tuning parameters of all the ELMo layers, we just learn these combination weights while training the model. All the word-level embeddings are concatenated to generate an enriched, contextual word-level representation $e$.

$$e = [\text{GloVE}(w), \text{CharEmbedding}(w), \text{ELMo}(w), \\ \text{POS-tag}(w), \text{NER-tag}(w), \text{position-tag}(w)] \quad (1)$$

### 3.3.2 Word-Level Cross-Attention Layer

To highlight the words in the conversation history $C_t$ relevant to the slot $s_i$, we add a word-to-word attention from conversation history to the slot. For computing the attention weights, we used symmetric scaled multiplicative attention [14] with a ReLU non-linearity. The weights are calculated according to equation 2 and used according to equation 3 to obtain the attended vector for each word in the conversation.

$$\alpha_{jk} = \frac{\exp(f(We_j^C)Df(We_k^s))}{\sum_{k=1}^{K}\exp(f(We_j^C)Df(We_k^s))} \quad (2)$$

$$\hat{a}_j^C = \sum_{k=1}^{K} \alpha_{jk} * e_k^s \quad (3)$$

Here, $e_j^C$ and $e_k^s$ correspond to the word embedding of the $j^{th}$ word in the conversation and $k^{th}$ word in the slot. The length of the slot is denoted by $K$. $f$ denotes a non-linear activation, which here is a ReLU. To get the representation $r_j$ for each word in the conversation history, we concatenate the attended vector with the initial word embedding: .

$$r_j^C = [e_j^C, \hat{a}_j^C] \quad (4)$$

For the slot representation for each word $k$ in the slot name, we use the word embedding $r_k^s = e_k^s$.

Note that symmetric scaled multiplicative attention with ReLU non-linearity is used in all attention computations of our proposed models, as we empirically found that it gives better performance compared to other attention variants.(e.g. multiplicative, scaled multiplicative, additive).

### 3.3.3 First Layer RNN

The computed representations $r_k^s$ and $r_j^C$ for each word in the slot name and the conversation history respectively, are then passed through a Gated Recurrent Unit (GRU) [6] in order to model the temporal interactions between the words and get a contextual representation. For each of $C_t$ and $s_i$, we use bidirectional GRUs and obtain the hidden contextual representation by averaging the hidden states of each GRU direction per time step:

$$H_1^C = [h_1^{C,1}, h_2^{C,1}, ..., h_J^{C,1}] = GRU([r_1^C, r_2^C, ...., r_J^C]) \quad (5)$$

$$H_1^s = [h_1^{s,1}, h_2^{s,1}, ..., h_K^{s,1}] = GRU([r_1^s, r_2^s, ...., r_K^s]) \quad (6)$$

Here, $H_1^C$ and $H_1^s$ are the sequences of encoded representations for the conversational history and slot name respectively, output by the first bidirectional GRU layer (assuming K is the slot name length and J the conversation history length in number of words).

### 3.3.4 Higher Level Cross Attention Layer

We add a cross-attention network on top of the the base RNN layer to attend over higher level representations generated by the previous RNN layer, i.e. $H_1^C$ and $H_1^s$. We used two-way cross-attention network, one from conversation history ($H_1^C$) to the slot ($H_1^s$) and the other in the opposite direction. This is inspired by several works in reading comprehension where cross attention is used to compute relations between a long passage and a query question [28, 4].

The **Slot to Conversation History attention sub-network** helps in highlighting the words in the conversation which are relevant to the slot for which we want to generate the value. Similar to the word level attention, the attention weights are calculated by equation 7.

$$\alpha_{jk} = \frac{\exp(f(Vh_j^{C,1})D'f(Vh_k^{s,1}))}{\sum_{j=1}^{J}\exp(f(Vh_j^{C,1})D'f(Vh_k^{s,1}))} \tag{7}$$

$$\hat{h}_k^{s,1} = \sum_{j=1}^{J}\alpha_{jk} * h_j^{C,1} \tag{8}$$

We fuse the attention vector $\hat{h}_k^{s,1}$ with it's corresponding hidden state $h_k^{s,1}$ for each word in the slot name as follows:

$$r_k^{s,1} = [h_k^{s,1}, \hat{h}_k^{s,1}, \hat{h}_k^{s,1} + h_k^{s,1}, \hat{h}_k^{s,1} * h_k^{s,1}] \tag{9}$$

where, $*$ is the element wise dot product operation.

Similarly, the **Conversation to Slot attention sub-network** computes attention weights to highlight which words in the slot name are most relevant to each word in the conversation history. This enriches the word representation in the conversation history $h_j^{C,1}$ with an attention based representation $\hat{h}_j^{C,1}$, resulting in a new representation $r_j^{C,1}$. All computations are similar as in the Slot to Conversation History attention, but in the reverse direction.

#### 3.3.5 Second Layer RNN

The representations $r_k^{s,1}$ and $r_j^{C,1}$ are then passed through a second bidirectional GRU layer, to obtain $h_k^{s,2}$ and $h_j^{C,2}$. This helps in fusing these vectors together along with the temporal information.

#### 3.3.6 Self Attention Layer

We add a self attention network on top of the conversation representation $h_j^{C,2}$. This layer helps resolve correlation between words across utterances in the conversation history. We introduce this sub-network to address cases where the user refers to slot values that are present in previous utterances, which is a common phenomenon in dialogs, especially multi-domain ones. Self attention is computed as:

$$\alpha_{ji} = \frac{\exp(f(Wh_j^{C,2})Df(Wh_i^{C,2}))}{\sum_{i=1}^{J}\exp(f(Wh_j^{C,2})Df(Wh_i^{C,2}))} \tag{10}$$

$$\hat{h}_j^{C,2} = \sum_{i=1}^{J}\alpha_{ji} * h_i^{C,2} \tag{11}$$

The final representation $r_j^{C,2}$ for each word in the conversation is the merged representation of self-attended vector $\hat{h}_j^{C,2}$ and the hidden state $h_j^{C,2}$, merged according to equation 9. We don't add self-attention for slot names as those are much shorter in length and we don't expect any co-reference for those cases.

#### 3.3.7 Third Layer RNN and Slot Summarization

We use a third layer RNN to get the final representation for the conversation history

$$h_j^{C,3} = GRU(r_j^{C,2}), j = 1..J \tag{12}$$

Since the slot name is much shorter in length than the conversation history, it can be encoded with less information. Instead of using an additional RNN, we summarize the slot using a linear transformation to reduce the slot representation into a single vector.

$$\alpha_k = w^{\intercal} * h_k^{s,2} \tag{13}$$

$$h^{s,3} = \sum_{k=1}^{K}\alpha_k * h_k^{s,2} \tag{14}$$

where, $w^\intercal$ is the parameter which is learnt during training.

Finally, $H^{C,3} = [h_1^{C,3}, h_2^{C,3}, ..., h_J^{C,3}]$ is the per word representation for the conversation history, while $h^{s,3}$ is the summarized slot name representation, both of which will be used at the decoding step.

### 3.4 Decoder and Slot Gate classifier

The decoder network is a GRU that decodes the value $v_i$ for slot $s_i$. At each decoding step $i$ that computes each word in the slot value, the network computes two distributions: a distribution over all in-vocabulary words (word generation distribution) and one over all words in the conversation history (word history distribution). This allows the decoder to generate unseen words that appear in the conversation history but are not present in the vocabulary of the training data. This formulation removes the dependency of having a predefined ontology that contains all the possible slot values, which is restrictive for free-form slots. Because of the ability to generate unseen (out-of-ontology) slot values, the network is well-suited for zero-shot use cases.

We initialize the decoder by combining the last hidden state of the conversation history representation and the summarized slot representation:

$$h_0^{dec} = W[h_J^{C,3}, h^{s,3}] \tag{15}$$

where W is a learnable parameter. At each decoding time-step $i$, the decoder generates a probability distribution over the vocabulary:

$$P_i^{vocab} = Softmax(W * h_i^{dec}) \tag{16}$$

The decoder also generates a probability distribution over words in the conversation history $P_{history}$ by using a pointer network [24], i.e., computing attention weights for each word in the conversation history.

To generate the final vocabulary distribution, we take a weighted sum of $P_{history}$ and $P_{vocab}$:

$$P_i^{final} = p_i^{gen} * P_i^{vocab} + (1 - p_i^{gen}) * P_i^{history} \tag{17}$$

Where $p_{gen}$ is the probability to generate a word as opposed to copy from the history, and is calculated at each decoder time step.

To avoid running the decoder for slots not present in the conversation, we also train a Slot Gate classifier[33]. This is a 3-way classifier which predicts among the following classes $\{none, dontcare, gen\}$. Only when the classifier predicts $gen$ we decode the slot value. When the classifier predicts "$none$" we assume that the slot is not present and takes a "$none$" value in the state, and when it predicts "$dontcare$", we assume the user does not care about the slot value (this appears commonly in dialog and therefore "$dontcare$" is a special value for DST systems).

The network is trained in a multi-task manner using standard cross entropy loss. We combine the losses of the slot generator (decoder) and the SG classifier as follows:

$$Loss_{combined} = Loss_{generator} + \gamma * Loss_{classifier} \tag{18}$$

where $\gamma$ is a hyperparameter that is optimized empirically.

## 4 Dataset

We evaluate our approach on MultiWOZ, a multi-domain Wizard-of-Oz dataset. MultiWOZ 2.0 is a recent dataset of labeled human-human written conversations spanning multiple domains and topics [3]. As of now, it is the largest labeled, goal-oriented, human-human conversational dataset with around 10k dialogs, each with an average of 13.67 turns. The data spans seven domains and 37 slot types. Due to patterns of annotation errors found in MultiWOZ 2.0, [8] re-annotated the data and released a MultiWOZ 2.1 version, which corrected a significant number of errors. Table 1 mentions the percentage of slots in each domain whose values changed with the MultiWOZ 2.1 re-annotation.

For all our experiments, we use MultiWOZ 2.1 data, which is shown to be cleaner and more challenging because many slots are now correctly annotated with their corrected values or $dontcare$ instead of $none$. We are using only five domains out of the available seven - namely $(\text{restaurant}, \text{hotel}, \text{attraction}, \text{taxi}, \text{train})$ - since the other two domains $(\text{bus}, \text{police})$ are only present in the training set.

| Slot Values Updated in MultiWOZ 2.1 | | | | | |
|---|---|---|---|---|---|
| | Restaurant | Taxi | Hotel | Train | Attraction |
| Train | 13.64 | 3.65 | 26.89 | 7.04 | 12.69 |
| Dev | 22.04 | 3.18 | 20.93 | 5.88 | 12.82 |
| Test | 19.33 | 3.95 | 24.70 | 10.59 | 16.12 |

Table 1: Percentage of slot values that changed in MultiWOZ 2.1 compared to MultiWOZ 2.0.

# 5 Evaluation

In this section we first describe the evaluation metrics and then present the results of our experiments.

To evaluate the performance of dialog state tracking models, two of the most common metrics are:

- **Average Slot Accuracy**: The average slot accuracy is defined as the fraction of slots for which the model predicts the correct slot value. It can be aggregated across all turns and dialogs.
- **Joint Goal Accuracy**: The joint goal accuracy is defined as the fraction of dialog turns for which the values $v_i$ for all slots $s_i$ are predicted correctly.

## 5.1 Experiment Details

| Single Domain | | | | |
|---|---|---|---|---|
| | MA-DST | | TRADE | |
| Domain | Joint | Slot | Joint | Slot |
| Hotel | 57.70 | 93.41 | 50.25 | 90.48 |
| Train | 76.47 | 94.87 | 74.47 | 94.30 |
| Taxi | 76.55 | 91.25 | 70.18 | 86.27 |
| Restaurant | 66.33 | 93.86 | 66.02 | 93.73 |
| Attraction | 72.49 | 89.38 | 68.48 | 86.89 |

Table 2: Joint goal and slot accuracy of MA-DST and TRADE on 5 single-domain datasets from MultiWOZ 2.1

We train the encoders to jointly optimize the losses of the slot gate classifier and the slot value generator decoder. The parameters of the model are shared for all $(domain, slot)$ pairs, which makes this model scalable to a large number of domain and slots. We train the model using stochastic gradient descent and use the Adam Optimizer. We empirically optimized the learning rate in the range $[0.0005 - 0.001]$ and used $0.0005$ for the final model, while we kept betas as $(0.9, 0.999)$ and epsilon 1x$10^{-08}$. We used a batch size of four dialog turns and for each turn we generate all 30 slot values. We decayed the learning rate after regular intervals (3 epochs) by a factor of $\theta$ $(0.25)$, which was empirically optimized. For ELMo, we kept a dropout of 0.5 for the contexual embedding and used $l_2$ regularization for the weights of ELMo. We used a dropout of 0.2 for all the layers everywhere else. For word embeddings, we used 300-dimensional GloVe embeddings and 100-dimensional character embeddings. For all the GRU and attention layers the hidden size is kept at 400. The weight $\gamma$ for the multi-task loss function in equation 18 is kept at 1.

## 5.2 Results

In this section, we present the results for our model. We measure the quality of the model on joint goal accuracy and average slot accuracy, as described earlier. As our baseline for comparison, we consider the TRADE model [33], which is the present state of the art for MultiWOZ. To have a fair comparison, we report the numbers on the corrected MultiWOZ 2.1 dataset for both models.

| Multi Domain | | |
|---|---|---|
| Model | Joint | Slot |
| Baseline (TRADE [33]) | 45.6 | 96.62 |
| Our Base Model | 44.0 | 96.15 |
| + Slot Gate + Word-Level Cross-Attention | 47.60 | 97.01 |
| + Higher-Level Cross-Attention | 49.56 | 97.15 |
| + Self-Attention + Slot Summarizer | 50.55 | 97.21 |
| + ELMo (MA-DST) | 51.04 | 97.28 |

Table 3: Joint goal and slot accuracy of different models in the all-domain setting of MultiWOZ 2.1

In Table 2, we present the results for DST on single-domain data. We create the train, dev, and test splits of the data for a particular domain by filtering for dialogs which only contain that domain. As shown in table 2, MA-DST outperforms TRADE for all five domains, improving the joint goal accuracy by up to 7% absolute as well as the average slot accuracy by up to 5% absolute.

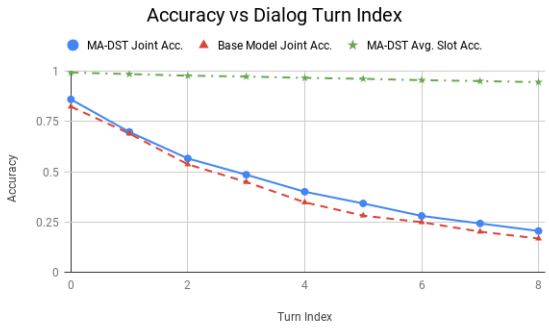


Figure 3: Per Slot Accuracy on test set in multi-domain setting for MA-DST.

| Zero Shot Experiment | | |
|---|---|---|
| | MA-DST | TRADE |
| Domain | Joint | Joint |
| Hotel | 16.28 | 14.20 |
| Train | 22.76 | 22.39 |
| Taxi | 59.27 | 59.21 |
| Restaurant | 13.56 | 12.59 |
| Attraction | 22.46 | 20.06 |

Figure 4: Joint goal accuracy of MA-DST and TRADE in the zero shot setting for the five domains of MultiWOZ 2.1.

Table 3 shows results for the multi-domain setting, where we combine all available domains during training and evaluation. We compare the the accuracy of MA-DST with the TRADE baseline and four additional ablation variants of our model. These four variants capture the contribution of the different sub-networks and layers in MA-DST on top of the base encoder-decoder architecture, which is called "Our Base Model" in Table 3. Our full proposed MA-DST model achieves the highest performance on joint goal accuracy and average slot accuracy, surpassing the current state-of-the-art performance. Each of the additional layers of self- and cross-attention contribute to progressively higher accuracy for both metrics.

In Table 4 we present the zero shot results. For these experiments, the test set contains only dialogs from the target domain while the training set contains only dialogs from the other four domains. As shown in Table 4, MA-DST outperforms TRADE's state-of-the-art result by up to 2% on the joint goal accuracy metric.

### 5.3 Error Analysis

In this section we study the errors being made by MA-DST. Figure 5 shows the per-slot accuracy in the all-domain setting, in descending order of performance. As seen from Figure 5, the MA-DST model tends to make the most errors for open-ended slots such as restaurant-name, attraction-name, hotel-name, train-leaveat. These slots are difficult to predict for the model because, unlike categorical slots, these slots can take on a large number of possible values and are more likely to encounter unseen values. On the other end of spectrum, we have slots like restaurant-bookday, hotel-bookstay, hotel-bookday, and train-day, for which the model is more than 99% accurate. As expected, most of the top-perfoming slots are categorical, i.e. they can take only a small number of different values from a pre-defined set.

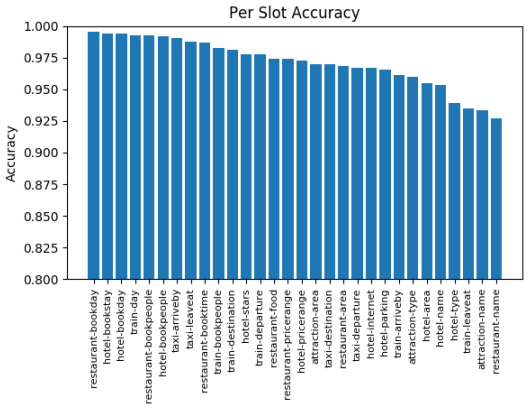


Figure 5: Per Slot Accuracy on test set in multi-domain setting for MA-DST.

Figure 3 analyzes the relationship between depth of conversation and accuracy of MA-DST. We bucket the dialog turns according to their turn index, and calculate the joint goal accuracy and average slot accuracy for each bucket. As shown in figure 3, the joint goal accuracy and average slot accuracy for MA-DST is around 88% and 99% for turn 0, and it decreases to 8% and 92% for turn 10. This shows that model's performance degrades as the conversation becomes longer. This can be explained by the fact that longer conversations tend to be more complex and can have longer-range dependencies. To study the effect of attention layers, we compare the joint goal accuracy of our base model and MA-DST for each turn. As can be seen from Figure 3, MA-DST performs better than our base model, which doesn't have the additional attention layers, for both earlier and later turns by an average margin of 4%.

## 6 Conclusion

We propose a new architecture for dialog state tracking that uses multiple levels of attention to better encode relationships between the conversation and slot semantics and resolve long-range cross-domain coreferences. It does not rely on knowing the ontology and both generate values from the vocabulary and copy values from the conversation history. It shares the same model weights for all $(domain, slot)$ pairs so it can easily be adapted to new domains and applied in a zero-shot or few-shot setting. MA-DST achieves 51% joint goal accuracy on the updated MultiWOZ 2.1. In the zero-shot setting we improve the state-of-the-art by over 2%.

## References


[1] Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*, 2017.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[3] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 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 (EMNLP)*, pages 5016–5026, 2018.

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

[5] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November 2016. Association for Computational Linguistics.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[7] 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*, pages 4171–4186, June 2019.

[8] Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. 2019.

[9] Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational ai. In *ACL and SIGIR tutorial*, July 2018. ACL and SIGIR tutorial.

[10] Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. Dialog state tracking: A neural reading comprehension approach, 2019.

[11] Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *arXiv preprint arXiv:1907.00883*, 2019.

[12] Matthew Henderson, Blaise Thomson, and Jason D. Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A., June 2014. Association for Computational Linguistics.

[13] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[14] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension, 2017.

[15] Yiping Kang, Yunqi Zhang, Jonathan K Kummerfeld, Lingjia Tang, and Jason Mars. Data collection for dialogue system: A startup perspective. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 33–40, 2018.

[16] Walter Stephen Lasecki, Ece Kamar, and Dan Bohus. Conversations in the crowd: Collecting data for task-oriented dialog learning. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[17] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000.

[18] Bing Liu and Ian Lane. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *Proc of Interspeech*, 2017.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. 2019.

[20] 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 (ACL) Volume 1: Long Papers*, pages 1777–1788, 2016.

[21] J. Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[22] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[23] Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. Scalable multi-domain dialogue state tracking. In *Proceedings of IEEE ASRU*, 2017.

[24] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

[25] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Comput. Speech Lang.*, 24(4):562–588, 2010.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, 2017.

[27] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc., 2015.

[28] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as possible but not simpler. *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 2017.

[29] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 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 (ACL): Volume 1, Long Papers*, pages 438–449, 2017.

[30] Jason Williams, Antoine Raux, and Matthew Henderson. The dialog state tracking challenge series: A review. *Dialogue and Discourse*, April 2016.

[31] Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference, The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 404–413, 2013.

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

[33] 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 *ACL*, 2019.

[34] 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*, pages 1448–1457, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[35] Chenguang Zhu, Michael Zeng, and Xuedong Huang. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv*, December 2018.