# Dialog State Tracking: A Neural Reading Comprehension Approach

**Shuyang Gao*, Abhishek Sethi*, Sanchit Agarwal***
**Tagyoung Chung, Dilek Hakkani-Tur**
Amazon Alexa AI
{shuyag, abhsethi, agsanchi, tagyoung, hakkanit}@amazon.com

## Abstract

Dialog state tracking is used to estimate the current belief state of a dialog given all the preceding conversation. Machine reading comprehension, on the other hand, focuses on building systems that read passages of text and answer questions that require some understanding of passages. We formulate dialog state tracking as a reading comprehension task to answer the question *what is the state of the current dialog?* after reading conversational context. In contrast to traditional state tracking methods where the dialog state is often predicted as a distribution over a closed set of all the possible slot values within an ontology, our method uses a simple attention-based neural network to point to the slot values within the conversation. Experiments on MultiWOZ-2.0 cross-domain dialog dataset show that our simple system can obtain similar accuracies compared to the previous more complex methods. By exploiting recent advances in contextual word embeddings, adding a model that explicitly tracks whether a slot value should be carried over to the next turn, and combining our method with a traditional joint state tracking method that relies on closed set vocabulary, we can obtain a joint-goal accuracy of 47.33% on the standard test split, exceeding current state-of-the-art by 11.75%**.

## 1 Introduction

A task-oriented spoken dialog system involves continuous interaction with a machine agent and a human who wants to accomplish a predefined task through speech. Broadly speaking, the system has four components, the Automatic Speech Recognition (ASR) module, the Natural Language Understanding (NLU) module, the Natural Language Generation (NLG) module, and the Dialog Manager. The dialog manager has two primary missions: dialog state tracking (DST) and decision making. At each dialog turn, the state tracker updates the belief state based on the information received from the ASR and the NLU modules. Subsequently, the dialog manager chooses the action based on the dialog state, the dialog policy and the backend results produced from previously executed actions.

Table 1 shows an example conversation with the associated dialog state. Typical dialog state tracking system combines user speech, NLU output, and context from previous turns to track what has happened in a dialog. More specifically, the dialog state at each turn is defined as a distribution over a set of predefined variables (Williams et al., 2005). The distributions output by a dialog state tracker are sometimes referred to as the tracker's belief or the belief state. Typically, the tracker has complete access to the history of the dialog up to the current turn.

Traditional machine learning approaches to dialog state tracking have two forms, generative and discriminative. In generative approaches, a dialog is modeled as a dynamic Bayesian network where true dialog state and true user action are unobserved random variables (Williams and Young, 2007); whereas the discriminative approaches are directly modeling the distribution over the dialog state given arbitrary input features.

Despite the popularity of these approaches, they often suffer from a common yet overlooked problem — relying on fixed ontologies. These systems, therefore, have trouble handling previously unseen mentions. On the other hand, reading comprehension tasks (Rajpurkar et al., 2016; Chen et al.,

*Authors contributed equally.
**We note that after publication, a new state-of-the-art can now be obtained with a similar attention mechanism followed by a enoder-decoder architecture (Wu et al., 2019).

2017; Reddy et al., 2019) require us to find the answer spans within the given passage and hence state-of-the-art models are developed in such a way that a fixed vocabulary for an answer is usually not required. Motivated by the limitations of previous dialog state tracking methods and the recent advances in reading comprehension (Chen, 2018), we propose a reading comprehension based approach to dialog state tracking. In our approach, we view the dialog as a *passage* and ask the question *what is the state of the current dialog?* We use a simple attention-based neural network model to find answer spans by directly pointing to the tokens within the dialog, which is similar to Chen et al. (2017). In addition to this attentive reading model, we also introduce two simple models into our dialog state tracking pipeline, a *slot carryover* model to help the tracker make a binary decision whether the slot values from the previous turn should be used; a *slot type* model to predict whether the answer is {`Yes`, `No`, `DontCare`, `Span`}, which is similar to Zhu et al. (2018). To summarize our contributions:

- We formulate dialog state tracking as a reading comprehension task and propose a simple attention-based neural network to find the state answer as a span over tokens within the dialog. Our approach overcomes the limitations of fixed-vocabulary issue in previous approaches and can generalize to unseen state values.

- We present the task of dialog state tracking as making three sequential decisions: i) a binary *carryover* decision by a simple slot carryover model ii) a *slot type* decision by a slot type model iii) a *slot span* decision by an attentive reading comprehension model. We show effectiveness of this approach.

- We adopt recent progress in large pre-trained contextual word embeddings, i.e., BERT (Devlin et al., 2018) into dialog state tracking, and get considerable improvement.

- We show our proposed model outperforms more complex previously published methods on the recently released MultiWOZ-2.0 corpus (Budzianowski et al., 2018; Ramadan et al., 2018). Our approach achieves a joint-goal accuracy of 42.12%, resulting in a 6.5% absolute improvement over previous state-of-the-art. Furthermore, if we combine our results with the traditional joint state tracking method in Liu and Lane (2017), we achieve a joint-goal accuracy of 47.33%, further advancing the state-of-the-art by 11.75%.

- We provide an in-depth error analysis of our methods on the MultiWOZ-2.0 dataset and explain to what extent an attention-based reading comprehension model can be effective for dialog state tracking and inspire future improvements on this model.

| | |
|---|---|
| User: | I need to book a hotel in the east that has 4 stars. |
| `Hotel` | `area=east, stars=4` |
| Agent: | I can help you with that. What is your price range? |
| User: | That doesn't matter if it has free wifi and parking. |
| `Hotel` | `parking=yes, internet=yes`<br>`price=dontcare, stars=4, area=east` |
| Agent: | If you'd like something cheap,<br>I recommend Allenbell |
| User: | That sounds good, I would also like a<br>taxi to the hotel from cambridge |
| `Hotel` | `parking=yes, internet=yes`<br>`price=dontcare, area=east, stars=4` |
| `Taxi` | `departure=Cambridge`<br>`destination=Allenbell` |

Table 1: An example conversation in MultiWOZ-2.0 with dialog states after each turn.

## 2 Related Work

**Dialog State Tracking** Traditionally, dialog state tracking methods assume a *fixed ontology*, wherein the output space of a slot is constrained by the predefined set of possible values (Liu and Lane, 2017). However, these approaches are not applicable for unseen values and do not scale for large or potentially unbounded vocabulary (Nouri and Hosseini-Asl, 2018). To address these concerns, a class of methods employing scoring mechanisms to predict the slot value from a endogenously defined set of candidates have been proposed (Rastogi et al., 2017; Goel et al., 2018). In these methods, the candidates are derived from either a predefined ontology or by extraction of a word or $n$-grams in the prior dialog context. Previously, Perez and Liu (2017) also formulated state tracking as a machine reading comprehension problem. However, their model architecture used a memory network which is relatively complex and still assumes a fixed-set vocabulary. Perhaps, the most similar technique to our work is the pointer networks proposed by Xu and Hu (2018) wherein an attention-based mechanism is employed to *point* the start and end token of a slot value. However, their formulation does not incorporate a *slot carryover* component and outlines an encoder-decoder architecture in which the slot type embeddings are derived from the last state of the RNN.

**Reading Comprehension** A reading comprehension task is commonly formulated as a supervised learning problem where for a given training dataset, the goal is to learn a predictor, which takes a passage $p$ and a corresponding question $q$ as inputs and gives the answer $a$ as output. In these tasks, an answer type can be cloze-style as in CNN/Daily Mail (Hermann et al., 2015), multiple choice as in MCTest (Richardson et al., 2013), span prediction as in SQuaD (Rajpurkar et al., 2016), and free-form answer as in NarrativeQA (Kočiský et al., 2018). In span prediction tasks, most models encode a question into an embedding and generate an embedding for each token in the passage and then a similarity function employing attention mechanism between the question and words in the passage to decide the starting and ending positions of the answer spans (Chen et al., 2017; Chen, 2018). This approach is fairly generic and can be extended to multiple choice questions by employing bilinear product for different types (Lai et al., 2017) or to free-form text by employing seq-to-seq models (Sutskever et al., 2014).

**Deep Contextual Word Embeddings** The recent advancements in the neural representation of words includes using character embeddings (Seo et al., 2016) and more recently using contextualized embeddings such as ELMO (Peters et al., 2018) and BERT (Devlin et al., 2018). These methods are usually trained on a very large corpus using a language model objective and show superior results across a variety of tasks. Given their wide applicability (Liu et al., 2019), we employ these architectures in our dialog state tracking task.

## 3 Our Approach

### 3.1 DST as Reading Comprehension

Let us denote a sub-dialog $D_t$ of a dialog $D$ as prefix of a full dialog ending with user's $t$th utterance, then state of the dialog $D_t$ is defined by the values of constituent slots $s_j(t)$, i.e., $S_t = \{s_1(t), s_2(t), .s_j(t), \ldots, s_M(t)\}$.

Using the terminology in reading comprehension tasks, we can treat $D_t$ as a *passage*, and for each slot $i$, we formulate a question $q_i$: *what is the value for slot i?* The dialog state tracking task then becomes understanding a sub-dialog $D_t$ and to answer the question $q_i$ for each slot $i$.

### 3.2 Encoding

**Dialog Encoding** For a given dialog $D_t$ at turn $t$, we first concatenate user utterances and agent utterances $\{\mathbf{u}_1, \mathbf{a}_1, \mathbf{u}_2, \mathbf{a}_2, \ldots, \mathbf{u}_t\}$. To differentiate between user utterance and agent utterance, we add symbol `[U]` before each user utterance and `[A]` before each agent utterance. Then, we use pre-trained word vectors to form $\mathbf{p}_i$ for each token in the dialog sequence and pass them as input into a recurrent neural network, i.e.,

$$\{\mathbf{d}_1, \mathbf{d}_2, \ldots \mathbf{d}_L\} = RNN(\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_L) \quad (1)$$

where $L$ is the total length of the concatenated dialog sequence and $\mathbf{d}_i$ is the output of RNN for each token, which is expected to encode context-aware information of the token. In particular, for pre-trained word vectors $\mathbf{p}_i$, we experiment with using deep contextualized word embeddings using BERT (Devlin et al., 2018). For RNN, we use a one layer bidirectional long short-term memory network (LSTM) and each $\mathbf{d}_i$ is the concatenation of two LSTMs from both directions, i.e., $\mathbf{d}_i = (\overleftarrow{\mathbf{d}_i}; \overrightarrow{\mathbf{d}_i})$. Furthermore, we denote $\mathbf{e}(t)$ as our dialog embedding at turn $t$ as follows:

$$\mathbf{e}(t) = (\overleftarrow{\mathbf{d}_1}; \overrightarrow{\mathbf{d}_L}) \quad (2)$$

**Question Encoding** In our methodology, we formulate questions $q_i$ defined earlier as *what is the value for slot i?* For each dialog, there are $M$ similar questions corresponding to $M$ slots, therefore, we represent each question $q_i$ as a fixed-dimension vector $\mathbf{q}_i$ to learn.

### 3.3 Models

**Overview** In our full model set up, three different model components are used to make a sequence of predictions: first, we use a *slot carryover* model for deciding whether to carryover a slot value from the last turn. If the first model decided not to carry over, a *slot type* model is executed to predict type of the answer from a set of {`Yes`, `No`, `DontCare`, `Span`}. If the *slot type* model predicts `span`, *slot span* model will finally be predicting the slot value as a span of tokens

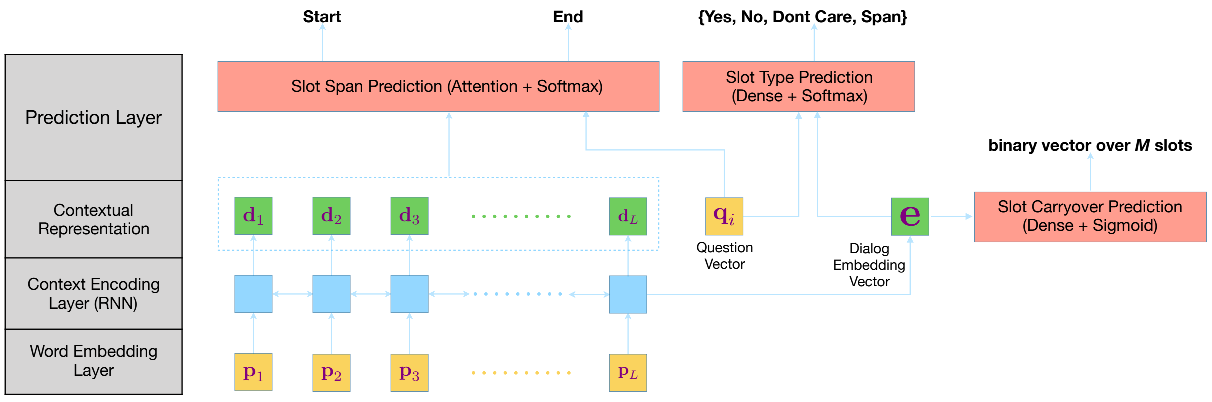
Start
End
{Yes, No, Dont Care, Span}
Prediction Layer
Slot Span Prediction (Attention + Softmax)
Slot Type Prediction (Dense + Softmax)
binary vector over M slots
Contextual Representation
d1
d2
d3
dL
qi
e
Slot Carryover Prediction (Dense + Sigmoid)
Question Vector
Dialog Embedding Vector
Context Encoding Layer (RNN)
Word Embedding Layer
p1
p2
p3
pL

Figure 1: Our attentive reading comprehension system for dialog state tracking. There are three prediction components on top (from right to left): 1) slot carryover model to predict whether a particular slot needs to be updated from previous turn 2) slot type model to predict the type of slot values from {`Yes`, `No`, `DontCare`, `Span`} 3) slot span model to predict the start and end span of the value within the dialog.

within the dialog. The full model architecture is shown in Figure 1.

**Slot Carryover Model** To model dynamic nature of dialog state, we introduce a model whose purpose is to decide whether to carry over a slot value from the previous turn. For a given slot $s_j$, $C_j(t) = 1$ if $s_j(t) \neq s_j(t-1)$ and 0 if they are equal. We multiply the dialog embedding $\mathbf{e}(t)$ with a fully connected layer $\mathbf{W}_i$ to predict the change for slot $i$ as:

$$P(C_i(t)) = sigmoid(\mathbf{e}(t) \cdot \mathbf{W}_i) \qquad (3)$$

The network architecture is shown in Figure 1. In our implementation, the weights $\mathbf{W}_i$ for each slot are trained together, i.e., the neural network would predict the slot carryover change $C_i(t)$ jointly for all $M$ slots.

**Slot Type Model** A typical dialog state comprises of slots that can have both categorical and named entities within the context of conversation. To adopt a flexible approach and inspired by the state-of-the-art reading comprehension approaches, we propose a classifier that predicts the type of slot value at each turn. In our setting, we prescribe the output space to be {`Yes`, `No`, `DontCare`, `Span`} where `Span` indicates the slot value is a named entity which can be found within the dialog. As shown in Figure 1, we concatenate the dialog embedding $\mathbf{e}(t)$ with the question encoding $\mathbf{q}_i$ for slot $i$ as the input to the affine layer $\mathbf{A}$ to predict the slot type $T_i(t)$ as:

$$P(T_i(t)) \propto \exp(\mathbf{A} \cdot (\mathbf{e}(t); \mathbf{q}_i)) \qquad (4)$$

**Slot Span Model** We map our slot values into a span with start and end position in our flattened conversation $D_t$. We then use the dialog encoding vectors $\{\mathbf{d}_1, \mathbf{d}_2, \dots \mathbf{d}_L\}$ and the question vector $\mathbf{q}_i$ to compute the bilinear product and train two classifiers to predict the start position and end position of the slot value. More specifically, for slot $j$,

$$P_j^{(start)}(x) = \frac{\exp\left(\mathbf{d}_x \Theta^{(start)} \mathbf{q}_j\right)}{\sum_{x'} \exp\left(\mathbf{d}_{x'} \Theta^{(start)} \mathbf{q}_j\right)} \qquad (5)$$

Similarly, we define $P_j^{(end)}(x)$ with $\Theta^{(end)}$. During span inference, we choose the best span from word $i$ to word $i'$ such that $i \leq i'$ and $P_j^{(start)}(i) \times P_j^{(end)}(i')$ is maximized, in line with the approach by Chen et al. (2017).

## 4 Experiments

### 4.1 Data

We use the recently-released MultiWOZ-2.0 dataset (Budzianowski et al., 2018; Ramadan et al., 2018) to test our approach. This dataset consists of multi-domain conversations from seven domains with a total of 37 slots across domains. Many of these slot types such as `day` and `people` are shared across multiple domains. In our experiments, we process each slot independently by considering the concatenation of slot domain, slot category, and slot name, e.g., {`bus.book.people`}, {`restaurant.semi.food`}. An example of conversation is shown in Table 1. We use standard training/development/test present in the data set.

It is worth-noting that the dataset in the current form has certain annotation errors. First, there is

| Method | Accuracy |
|---|---|
| MultiWOZ Benchmark | 25.83% |
| GLAD (Zhong et al., 2018) | 35.57% |
| GCE (Nouri and Hosseini-Asl, 2018) | 35.58% |
| Our approach (single) | 39.41% |
| Our approach (ensemble) | 42.12% |
| HyST (ensemble) (Goel et al., 2019) | 44.22% |
| **Our approach + JST (ensemble)** | **47.33%** |

Table 2: Joint goal accuracy on MultiWOZ-2.0. We present both single and ensemble results for our approach.

lack of consistency between the slot values in the ontology and the ground truth in the context of the dialog. For example, the ontology has *moderate* but the dialog context has *moderately*. Second, there are erroneous delay in the state updates, sometimes extending turns in the dialog. This error negatively impacts the performance of the slot carryover model.

### 4.2 Experimental Setup

We train our three models independently without sharing the dialog context. For all the three models, we encode the word tokens with BERT (Devlin et al., 2018) followed by an affine layer with 200 hidden units. This output is then fed into a one-layer bi-directional LSTM with 50 hidden units to obtain the contextual representation as show in Figure 1. In all our experiments, we keep the parameters of the BERT embeddings frozen.

For slot carryover model, we predict a binary vector over 37 slots jointly to get the decisions of whether to carry over values for each slot. For slot type and slot span models, we treat dialog–question pairs $(D_t, q_i)$ as separate prediction tasks for each slot.

We use the learning rate of 0.001 with ADAM optimizer and batch size equal to 32 for all three models. We stop training our models when the loss on the development set has not been decreasing for ten epochs.

## 5 Results

Table 2 presents our results on MultiWOZ-2.0 test dataset. We compare our methods with global-local self-attention model (GLAD) (Zhong et al., 2018), global-conditioned encoder model (GCE) (Nouri and Hosseini-Asl, 2018), and hybrid joint state tracking model (OV ST+JST) (Liu and Lane, 2017; Goel et al., 2019). As in previous work, we report joint goal accuracy as our metric. For each user turn, joint goal accuracy checks whether all predicted states exactly matches the ground truth state for all slots. We can see that our system with single model can achieve 39.41% joint goal accuracy, and with the ensemble model we can achieve 42.12% joint goal accuracy.

Table 3 shows the accuracy for each slot type for both our method and the joint state tracking approach with fix vocabulary in Goel et al. (2019). We can see our approach tends to have higher accuracy on some of the slots that have larger set of possible values such as `attraction.semi.name` and `taxi.semi.destination`. However, it is worth-noting that even for slots with smaller vocabulary sizes such as `hotel.book.day` and `hotel.semi.pricerange`, our approach achieves better accuracy than using closed vocabulary approach. Our hypothesis for difference is that such information appear more frequently in user utterance thus our model is able to learn it more easily from the dialog context.

We also reported the result for a hybrid model by combining our approach with the JST approach in (Goel et al., 2019). Our combination strategy is as follows: first we calculated the slot type accuracy for each model on the development dataset; then for each slot type, we choose to use the predictions from either our model or JST model based on the accuracy calculated on the development set, whichever is higher. With this approach, we achieve the joint-goal accuracy of 46.28%. We hypothesize that this is because our method uses an open vocabulary, where all the possible values can only be obtained from the conversation; the joint state tracking method uses closed ontology, we can get the best of both the worlds by combining two methods.

### 5.1 Ablation Analysis

Table 4 illustrates the ablation studies for our model on development set. The contextual embedding BERT (Devlin et al., 2018) can give us around 2% gains. As for the oracle models, we can see that even if using all the oracle results (ground truth), our development set accuracy is only 73.12%. This is because our approach is only considering the values within the conversation, if values are not present in the dialog, the

| Slot Name | Ours | JST | Vocab Size |
|---|---|---|---|
| attraction.semi.area | 0.9637 | 0.9719 | 16 |
| **attraction.semi.name** | 0.9213 | 0.9013 | 137 |
| attraction.semi.type | 0.9205 | 0.9637 | 37 |
| bus.book.people | 1.0000 | 1.0000 | 1 |
| bus.semi.arriveBy | 1.0000 | 1.0000 | 1 |
| bus.semi.day | 1.0000 | 1.0000 | 2 |
| bus.semi.departure | 1.0000 | 1.0000 | 2 |
| bus.semi.destination | 1.0000 | 1.0000 | 5 |
| bus.semi.leaveAt | 1.0000 | 1.0000 | 2 |
| **hospital.semi.department** | 0.9991 | 0.9988 | 52 |
| **hotel.book.day** | 0.9863 | 0.9784 | 11 |
| hotel.book.people | 0.9714 | 0.9847 | 9 |
| hotel.book.stay | 0.9736 | 0.9809 | 9 |
| **hotel.semi.area** | 0.9679 | 0.9570 | 24 |
| hotel.semi.internet | 0.9713 | 0.9718 | 8 |
| **hotel.semi.name** | 0.9147 | 0.9056 | 89 |
| hotel.semi.parking | 0.9563 | 0.9657 | 8 |
| **hotel.semi.pricerange** | 0.9679 | 0.9666 | 9 |
| hotel.semi.stars | 0.9627 | 0.9759 | 13 |
| hotel.semi.type | 0.9140 | 0.9261 | 18 |
| **restaurant.book.day** | 0.9874 | 0.9871 | 10 |
| restaurant.book.people | 0.9787 | 0.9881 | 9 |
| **restaurant.book.time** | 0.9882 | 0.9578 | 61 |
| restaurant.semi.area | 0.9607 | 0.9654 | 19 |
| **restaurant.semi.food** | 0.9741 | 0.9691 | 104 |
| **restaurant.semi.name** | 0.9113 | 0.8781 | 183 |
| **restaurant.semi.pricerange** | 0.9662 | 0.9626 | 11 |
| **taxi.semi.arriveBy** | 0.9893 | 0.9719 | 101 |
| **taxi.semi.departure** | 0.9665 | 0.9304 | 261 |
| **taxi.semi.destination** | 0.9634 | 0.9288 | 277 |
| **taxi.semi.leaveAt** | 0.9821 | 0.9524 | 119 |
| train.book.people | 0.9586 | 0.9718 | 13 |
| **train.semi.arriveBy** | 0.9738 | 0.9491 | 107 |
| **train.semi.day** | 0.9854 | 0.9783 | 11 |
| train.semi.departure | 0.9599 | 0.9710 | 35 |
| train.semi.destination | 0.9538 | 0.9699 | 29 |
| **train.semi.leaveAt** | 0.9595 | 0.9478 | 134 |

Table 3: Slot accuracy breakdown for our approach versus joint state tracking method. Bolded slots are the ones have better performance using our attentive reading comprehension approach.

oracle models would fail. It is interesting to see that if we replace our slot carryover model with an oracle one, the accuracy improves significantly to 60.18% (+19.08%) compared to replacing other two models (41.43% and 45.77%). This is because our span-based reading comprehension approach model already gives us accuracy as high as 96% per slot on development data, there is not much room for improvement. Whereas our binary slot carryover model only achieve an accuracy of 72% per turn. We hypothesis that for slot carryover problem is imbalanced, i.e., there are significantly more slot carryovers than slot updates, making the model training and predictions harder. This suggest further improvements are needed for *slot carryover* model to make overall state tracking accuracy higher.

| Ablation | Dev Accuracy |
|---|---|
| Oracle Models | 73.12% |
| Our approach | 41.10% |
| - BERT | 39.19% |
| + Oracle Slot Type Model | 41.43% |
| + Oracle Slot Span Model | 45.77% |
| + Oracle Slot Carryover Model | 60.18% |

Table 4: Ablation study on our model components for MultiWOZ-2.0 on development set for joint goal accuracy.

## 5.2 Error Analysis

In Table 5, we conduct an error analysis of our models and investigate its performance for different use cases. Since we formulate the problem to be an open-vocabulary state tracking approach wherein the slot values are extracted in the dialog context, we divide the errors into following categories:

- **Unanswerable Slot Error** This category contains two type of errors: (1) Ground truth slot is a not *None* value, but our prediction is *None*; (2) Ground truth slot is *None*, but our prediction is a not *None* value. This type of error can be attributed to the incorrect predictions made by our slot carryover model.

- **Imprecise Slot Reference** where multiple potential candidates in the context exists. The model refers to the incorrect *entity* in the conversation. This error can be largely attributed to following reasons: (1) the model overfits to the set of tokens that it has seen more frequently in the training set; (2) the model does not generalize well for scenarios where the user corrects the previous entity; (3) the model incorrectly overfits to the order or position of the entity in the context. These reasons motivate future research in incorporating more neural reading comprehension approaches for dialog state tracking.

- **Imprecisie Slot Resolution** In this type of errors, we cannot find the exact match of ground truth value in the dialog context.

| Category | Hypothesis | Reference | Context | (%) |
|---|---|---|---|---|
| Unanswerable Slot Error | not *None* | *None* | . . . | 42.4 |
| | *None* | not *None* | . . . | 23.1 |
| Incorrect slot Reference | 4 | 8 | . . . 3 nights, and **4 people**. Thank You! [A] Booking was unsuccessful . . . I'd like to book there Monday for 1 night with **8 people**. . . . | 19.1 |
| Incorrect Slot Resolution | 3:30 | 15:30 | . . . you like to arrive at the Cinema? [U] I want to leave the hotel by **3:30** [A] Your taxi reservation departing . . . | 12.9 |
| Imprecise Slot Boundary | nandos city centre | nandos | . . . number is 01223902168 [U] Great I am also looking for a restaurant called **nandos city centre** . . . | 2.5 |

Table 5: Error categorization and percentage distribution: representative example from each category and an estimate breakdown of the error types on development set, based on the analysis of 200 error samples produced by our model. Numbers of the first category is exact because we are able to summarize this error category statistically.

However, our predicted model span is a paraphrase or has very close meaning to the ground truth. This error is inherent in approaches that do not extract the slot value from an ontology but rather the dialog context. On similar lines, we also observe cases where the slot value in the dialog context is *resolved* (or canonicalized) to a different surface-form entity that is perhaps more amenable for downstream applications.

- **Imprecise Slot Boundary** In this category of errors, our model chooses a span that is either a superset or subset of the correct reference. This error is especially frequent for proper nouns where the model has a weaker signal to outline the slot boundary precisely.

Table 5 provides us the error examples and estimated percentage from each category. "Unanswerable Slot" accounts for 65.5% errors for our model, this indicates further attention may be needed to the slot carryover model, otherwise it would become a barrier even if we have a perfect span model. This finding is in alignment with our ablation studies in Table 4, where oracle slot carryover model would give us the most boost in joint goal accuracy. Additionally, 12.9% of errors are due to imprecise slot resolution, this suggests future directions of resolving the context words to the ontology.

## 5.3 Evaluating Different Context Encoders for Slot Carryover Model

As shown in oracle ablation studies in Table 4, slot carryover model plays a significant role in our pipeline. Therefore we explore the different types of context encoders for slot carryover model to see whether if it improves the performance in table 6. In addition to use a flat dialog context of user and agent turns `[U]` and `[A]` to predict carryover for every slot in the state, we explored hierarchical context encoder with an utterance-level LSTM over each user and agent utterance and a dialog-level LSTM over the whole dialog with both constrained and unconstrained context window, similar to Liu and Lane (2017). However, we did not witness any significant performance change across the two variants as show in Table 6. Lastly, we employed self-attention over the flattened dialog context in line with Vaswani et al. (2017). However, we can see from Table 6 that this strategy slightly hurts the model performance. One hypothesis for sub par slot carryover model performance is due to the inherent noise in the annotated data for state updates. Through a preliminary analysis on the development set, we encountered few erroneous delay in the state updates sometimes extending to over multiple turns. Nevertheless, these experimental results motivate future research in *slot carryover* models for multi-domain conversations.

## 5.4 Analyzing Conversation Depth

In Table 7, we explore the relationship between the depth of a conversation and the performance of our models. More precisely, we segment a given set of dialogs into individual *turns* and measure the state accuracy for each of these segments. We mark a turn correct only if all the slots in its state are predicted correctly. We observe that the model perfor-

| Context Feature | Per Turn Carryover Accuracy |
|---|---|
| Flat Context (LSTM) | 75.10% |
| Hierarchical Context (all turns) | 75.98% |
| Hierarchical Context ($\leq$ 3 turns) | 75.60% |
| Flat Context (Self-Attention) | 74.75% |

Table 6: Analyzing the different types of context features for Slot Carryover Model

| Conversation Depth $t$ | Total Turns | % Incorrect Turns |
|---|---|---|
| 1 | 1000 | 23.90 |
| 2 | 1000 | 38.30 |
| 3 | 997 | 50.85 |
| 4 | 959 | 61.52 |
| 5 | 892 | 71.52 |
| 6 | 811 | 76.82 |
| 7 | 656 | 82.77 |
| 8 | 475 | 87.37 |
| 9 | 280 | 89.64 |
| 10 | 153 | 94.77 |

Table 7: Analyzing the overall model robustness for conversation depth for MultiWOZ-2.0

mance degrades as the number of turns increase. The primary reason for this behavior is that an error committed earlier in the conversation can be carried over for later turns. This results in a strictly higher probability for a later turn to be incorrect as compared to the turns earlier in the conversation. These results motivate future research in formulating models for state tracking that are more robust to the depth of the conversation.

## 6 Conclusion

The problem of tracking user's belief state in a dialog is a historically significant endeavor. In that context, research on dialog state tracking has been geared towards discriminative methods, where these methods are usually estimating the distribution of user state over a fixed vocabulary. However, modern dialog systems presents us with problems requiring a large scale perspective. It is not unusual to have thousands of slot values in the vocabulary which could have millions variations of dialogs. So we need a vocabulary-free way to pick out the slot values.

How can we pick the slot values given an infinite amount of vocabulary size? Some methods adopt a candidate generation mechanism to generate slot values and make a binary decision with the dialog context. Attention-based neural network gives a clear and general basis for selecting the slot values by direct pointing to the context spans. While this type of methods has already been proposed recently, we explored this type of idea furthermore on MultiWOZ-2.0 dataset.

We introduced a simple attention based neural network to encode the dialog context and point to the slot values within the conversation. We have also introduced an additional slot carryover model and showed its impact on the model performance. By incorporating the deep contextual word embeddings and combining the traditional fixed vocabulary approach, we significantly improved the joint goal accuracy on MultiWOZ-2.0.

We also did a comprehensive analysis to see to what extent our proposed model can achieve. One interesting and significant finding from the oblation studies suggests the importance of the slot carryover model. We hope this finding can inspire future dialog state tracking research to work towards this direction, i.e., predicting whether a slot of state is none or not.

The field of machine reading comprehension has made significant progress in recent years. We believe human conversation can be viewed as a special type of context and we hope that the developments suggested here can help dialog related tasks benefit from modern reading comprehension models.

## Acknowledgements

The authors would like to thank Rahul Goel, Anuj Kumar Goyal, Angeliki Metallinou and other Alexa AI team members for their useful discussions and feedbacks.

## References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 2018. 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)*.

Danqi Chen. 2018. *Neural Reading Comprehension and Beyond*. Ph.D. thesis, Stanford University.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Rahul Goel, Shachi Paul, Tagyoung Chung, Jeremie Lecomte, Arindam Mandal, and Dilek Hakkani-Tur. 2018. Flexible and scalable state tracking framework for goal-oriented dialogue systems. *arXiv preprint arXiv:1811.12891*.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *Proc. Interspeech 2017*, pages 2506–2510.

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

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

Julien Perez and Fei Liu. 2017. 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*, pages 305–314.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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

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

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

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association of Computational Linguistics (TACL)*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jason D Williams, Pascal Poupart, and Steve Young. 2005. Factored partially observable markov decision processes for dialogue management. In *Proc. IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 76–82.

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

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

Puyang Xu and Qi Hu. 2018. 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 (ACL)*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *arXiv preprint arXiv:1805.09655*.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*.