Multi-Domain Dialogue State Tracking By Neural-Retrieval Augmentation

Lohith Ravuru¹, Seonghan Ryu², Hyungtak Choi¹, Haehun Yang¹, Hyeonmok Ko¹

¹Samsung Research, Seoul, South Korea
²42dot, Seoul, South Korea
loki.ravuru,ht777.choi,haehun.yang,felix.ko@samsung.com
seonghan.ryu@42dot.ai

Abstract

Dialogue State Tracking (DST) is a very complex task that requires precise understanding and information tracking of multi-domain conversations between users and dialogue systems. Many task-oriented dialogue systems use dialogue state tracking technology to infer users’ goals from the history of the conversation. Existing approaches for DST are usually conditioned on previous dialogue states. However, the dependency on previous dialogues makes it very challenging to prevent error propagation to subsequent turns of a dialogue. In this paper, we propose Neural Retrieval Augmentation to alleviate this problem by creating a Neural Index based on dialogue context. Our NRA-DST framework efficiently retrieves dialogue context from the index built using a combination of unstructured dialogue state and structured user/system utterances. We explore a simple pipeline resulting in a retrieval-guided generation approach for training a DST model. Experiments on different retrieval methods for augmentation show that neural retrieval augmentation is the best performing retrieval method for DST. Our evaluations on the large-scale MultiWOZ dataset show that our model outperforms the baseline approaches.

1 Introduction

Dialogue State Tracking (DST) involves analyzing the user’s dialogue and previous turn state expressed during the conversation, extracting the user’s goal/intent, and representing it in the form of a well-defined set of slots and values (Williams et al., 2016; Henderson, 2015; Williams and Young, 2007; Gao et al., 2018). The release of a large-scale multi-domain conversational data set (MultiWOZ Budzianowski et al., 2018) prompted advances in cross-domain dialogue systems. Figure 1 shows an example from the dataset where the user starts the conversation about reserving a hotel, then requests for booking a taxi, and finally, changes the original hotel reservation. The dialogue state here is defined as list of \texttt{<slot-value>} pairs for each \texttt{[domain]} (e.g., \texttt{[hotel] people 2 stay 5 days}), \texttt{[taxi] departure Hotel Santa}).

Recent works approach this either by classifying each slot over pre-defined slot-values that are selected from an ontology based on training data (Ma et al., 2019; Li et al., 2020) or first classifying a slot and then detecting the span of text in the original context as value for that slot (Kim et al., 2020; Gao et al., 2019). However, these models are highly dependant on the values in the dataset and the ontology. Another approach to DST is generating the value of a slot or both slot and value using a sequence-to-sequence model (Wu et al., 2019; Le et al., 2020). Papers using large pre-trained models such as GPT2 (Radford et al., 2019) have shown promising results (Budzianowski and Vulic, 2019; Hosseini-Asl et al., 2020). A single generative model can also be used to manage entire dialogue...
Figure 2: Different steps involved in the NRA-DST approach. The Query Encoder and Key Encoder are trained together. Once trained, Key Encoder is used to create a neural index and Query Encoder is used for retrieving results which are used in finetuning the T5 Model (Raffel et al., 2020), a pretrained Language Model which is used as backbone for our model.

by generating dialogue state, system action, and user response altogether (Lin et al., 2020; Hosseini-Asl et al., 2020). But these models are more prone to error propagation as explained below.

Dialogue State can be considered as a representation of the entire conversation and is used by subsequent modules in resolving system’s action and response. Error in the dialogue state propagates not only to these other modules but also to dialogue states of subsequent turns. To analyze this issue, we perform a simple analysis similar to Kim et al. (2020), by replacing the previous dialogue state with ground truth on the state-of-the-art MinTL (Lin et al., 2020) model. As shown in Table 1, using ground truth previous dialogue state in place of the generated previous dialogue state creates a difference of 27% in the prediction of current dialogue state. To bridge the performance gap and reduce error propagation, we propose augmenting retrieved dialogue states of similar dialogue contexts from a pre-computed index.

<table>
<thead>
<tr>
<th>Predicted Dialogue State</th>
<th>Actual Dialogue State</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinTL (T5-small)</td>
<td>51.0</td>
</tr>
<tr>
<td>MinTL (T5-base)</td>
<td>51.4</td>
</tr>
</tbody>
</table>

Table 1: Analysis of Error Propagation in MinTL model.

Large pre-trained models have shown to be very efficient in retrieval-based approaches compared to sparse representations based on TF/IDF, or BM25 (Guu et al., 2020; Lee et al., 2019; Karpukhin et al., 2020). Several works in open domain question answering have augmented retrieval-based results for better response generation (Lewis et al., 2020). However, this is generally done on natural text such as a question or a passage. In Thulke et al. (2021), the retrieval is done using an unstructured dialogue state, but the index is created only from structured paragraph text data.

In this work, we aim to improve DST by leveraging Neural Retrieval-Augmentation on a combination of unstructured dialogue state and structured user/system utterances.

The contributions of our work are as follows:

- We propose an NRA-DST framework that utilizes state-of-the-art neural retrieval methods and integrates it to Dialogue State Tracking for more efficient task-oriented conversations.
- We evaluate our framework on MultiWOZ 2.0 dataset and show that neural retrieval augmentation improves the performance.
- We conduct a comprehensive ablation analysis showing the effectiveness of our proposed framework.

2 Background

In this section, we briefly explain the notations used in further sections. Let us denote the dialogue with \( t \) turns as, \( D = \{(u_1, r_1), (u_2, r_2), \ldots (u_t, r_t)\} \), where \( u_i \) represents user utterance at \( i^{th} \) turn and \( r_i \) represents system response at \( i^{th} \) turn. Over the course of a dialogue, the goal of DST is to keep track of a dialogue state, \( dst = \{(d_1, (s_1, v_1)), (s_2, v_2), \ldots, (d_k, (s_1, v_1))\} \) where \( d_k \) is the domain, \( s_i \) is a slot from the domain, \( d_k \) and \( v_i \) is the value of \( s_i \). The dialogue context
at turn $t$ is defined as, $c_t = (dst_{t-1}, u_{t-1}, r_{t-1}, u_t)$.
In this paper, we formulate dialogue context using only the last turn but this can be extended to multiple previous turns. We formulate the original DST task as predicting the dialogue state from the dialogue context, $dst_t = model(c_t)$.

The concept of Belief Span (Lei et al., 2018) allows dialogue states to be represented as a span of text, enabling the conversion of a classification problem into a generation problem. Lin et al. (2020) builds upon belief spans and defines Levenshtein Belief Span ($lev_t$) as a minimal editing from previous dialogue state $dst_{t-1}$ to current dialogue state $dst_t$. For example,

$$dst_{t-1} \leftarrow \text{[restaurant] food french, price cheap, day Sunday}$$

$$dst_t \leftarrow \text{[restaurant] food thai, day Sunday, area centre}$$

$$lev_t \leftarrow dst_t - dst_{t-1}$$

$$lev_t = \text{[restaurant] food thai, day NULL, area centre}$$

We extend the belief spans by creating a neural index and guiding the model with possible Levenshtein spans from the retrieved result. The retrieved topk result contains possible DST updates, $lev_{1:k}$.

$$dst_t = NIRDST(lev_{1:k}, c_t)$$

The DST task is now updated as predicting the dialogue state from a combination of retrieved results and dialogue context as in Eq 1. Figure 2 describes the architecture of NRA-DST.

### 3 Methods

Given a training dataset $D_{train} = \{D_1, D_2, ..., D_n\}$, we create a neural index, $D_{index}$ such that we can query the index based on neural representation (latent space representation) of dialogue context $c_t$, which is a combination of previous dialogue state $dst_{t-1}$ and user/system utterances. Section 3.1 explains the $D_{index}$ creation method in detail. The contents of the $D_{index}$ can be represented as $(E(c_t), lev_t)$, where the key, $E(c_t)$ is the neural representation of dialogue context and the value, $lev_t$ represents the corresponding dialogue state updates. The key idea is that given a dialogue context, we retrieve domains and slots detected in another dialogue with a similar context. Figure 2 shows an example of similar contexts, $c_i$ and $c_i^+$. The previous dialogue states of both contexts contain the slots named "area" and "stars", from the domain named "hotel" and the utterances are also similar.

### 3.1 Neural Dialogue Context Retrieval

For generating efficient Neural Representations, we use a modification of the state-of-the-art Dense Passage Retrieval (DPR) Model (Karpukhin et al., 2020). Similar to the dual-encoder approach proposed in the DPR model, we use two different encoders: Query Encoder ($E_q$) and Key Encoder ($E_k$). The DPR model is trained so that the dot-product similarity (Eq 2) is higher for similar dialogue contexts.

$$sim(c_i, c_j) = E_q(c_i)^T E_k(c_j)$$

Training for the similarity metric 2 requires labelling the dataset with positive and negative contexts. For each turn of the dialogue in the training corpus of the original MultiWOZ dataset, we use a customized Algorithm 1 to generate a positive context ($c_i^+$) and a negative context ($c_i^-$).

#### Algorithm 1: Creating Training Data for fine-tuning DPR model.

```python
1 def PrepareTrainingInstance:
2   Input : Dialogue Context ($U_i$)
3   Output: Positive Dialogue Context ($U_i^+$), Negative Dialogue Context ($U_i^-$)
4   Similar Context, $U_{bm25}[100] \leftarrow$ BM25 top100 results from training data;
5   $Q \leftarrow \{\}$;
6   $lev_i \leftarrow dst(U_i) - previous_dst(U_i)$;
7   foreach dialogue context $U_j \in U_{bm25}$ do
8     $lev_j \leftarrow dst(U_j) - previous_dst(U_j)$;
9     $score \leftarrow slot_F1(lev_i, lev_j)$;
10    $Q.append((score, U_j))$;
11 end
12 $sort(Q, key a : a[0])$;
13 $U_i^+, U_i^- \leftarrow Q[0][1], Q[99][1]$;
14 return $U_i^+, U_i^-$;
```

Due to limitations of memory and training time with RoBERTa-base as encoder, we limit the positive and negative contexts to only one context each. We also perform the original DPR model’s optimization trick of using in-batch negatives to train effectively. Although we used Algorithm 1 to select only one negative context for a particular training instance, positive contexts from other training
instances in a single training batch are also considered as negative contexts for that instance.

\[
L(c_i, c_i^+, \ldots, c_i^{-n}) = - \log \left( \frac{e^{sim(c_i, c_i^+)}}{e^{sim(c_i, c_i^+) + \sum_{k=1}^{n} e^{sim(c_i, c_i^{-k})}} } \right) 
\]  

(3)

After training the model with the loss function 3, the Key Encoder is used to create the neural index, whereas the Query Encoder is used along with the Dialogue State Tracking model for retrieving the result.

3.2 Generation based Dialogue State Tracking

The retrieval result from Neural Index \((lev_{topk})\) is appended to the original dialogue context \(c_t\), as described in Eq 1. All sequences are concatenated by using special end-of-sequence (eos) tokens to form a single retrieval-augmented context \((c^*_t)\) and given as input to the T5 (Raffel et al., 2020) encoder.

\[
c^*_t \leftarrow lev_1 \langle eos_l1 \rangle \; lev_2 \langle eos_l2 \rangle \ldots \; dst_{t-1} \langle eos_b \rangle \; rt_{t-1} \langle eos_r \rangle \; ut_t \langle eos_u \rangle
\]

\[H = \text{Encoder}(c^*_t)\]  

(4)

The T5 decoder model takes as input the encoder hidden states and generates updates to the dialogue state.

\[lev_t = \text{Decoder}(H)\]  

(5)

The loss function used in the Dialogue State Generation model is standard negative loss-likelihood between the ground truth \(lev_t\) and generated \(lev_t\). The final dialogue state, \(dst_t\) is derived by combining \(lev_t\) and \(dst_{t-1}\).

4 Experiments

4.1 Datasets

We evaluate our framework on the Multi-Domain Wizard-of-Oz (MultiWOZ 2.0) (Budzianowski et al., 2018) dataset. The dataset consists of various human-to-human conversations, including tasks from seven different domains (restaurant, train, attraction, hotel, taxi, hospital, police). We used the original dataset split with a training corpus of 8438 dialogues, a validation corpus of 1000 dialogues, and a test corpus of 1000 dialogues.

4.2 Experimental Setup

We implemented our proposed methods on top of the code from MinTL framework (Lin et al., 2020) and Dense-Passage Retrieval model (Karpukhin et al., 2020). For BM25, we use the implementation from Pyserini (pys). We use approximate nearest neighbours with the FAISS library (fai) for performing our retrieval from the neural index. All the hyperparameters used are the default parameters from the baseline implementations.

For our retrieval model, we use RoBERTa (Liu et al., 2019) for Key Encoder, Value Encoder and we use T5-small as the backbone for our DST model. We trained our retrieval model and created the neural index with only the training corpus of the original dataset.

4.3 Metrics

**Joint Goal Accuracy** measures the accuracy of the generated DST by comparing them to the ground truth DST. The generated slot-value is considered accurate only if it is exactly matching the ground truth slot-value. The accuracy is calculated over each turn for dialogue, and it is averaged over the entire dialogue.

**Slot Detection Error** is a custom metric that evaluates the benefit of Retrieval Augmentation. It is the error in the ground truth DST and generated DST, but the exact value of the slot is not matched.

4.4 Results

Table 2 describes results on our NRA-DST model compared to other retrieval methods. We compare our model with other generation based baselines DSTQA (Zhou and Small, 2019), NADST (Le et al., 2020), SOM-DST (Kim et al., 2020). We also compare our model with our custom retrieval baselines.

**BM25-Retrieval DST Model** uses bm25, bag-of-words, retrieval algorithm to create the neural index and retrieve the top-k results.

**RoBERTa-Retrieval DST Model** uses a pre-trained RoBERTa model directly without any fine-tuning for creating the index.

The decrease in Slot Detection Error and an increase in Joint Goal Accuracy shows that augmenting retrieval results is beneficial for generation-based DST models. We observe that our proposed NRA-DST method outperforms all other retrieval-based models.
### Table 2: Results on MultiWOZ 2.0 dataset compared to different baselines. *: results reported by the original paper. †: Uses T5-small model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Accuracy (↑)</th>
<th>Slot Detection Error (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSTQA (Zhou and Small, 2019)*</td>
<td>51.44</td>
<td>-</td>
</tr>
<tr>
<td>NADST (Le et al., 2020)*</td>
<td>50.52</td>
<td>-</td>
</tr>
<tr>
<td>SOMDST (Bert-base) (Kim et al., 2020)*</td>
<td>51.72</td>
<td>-</td>
</tr>
<tr>
<td>MinTL (Lin et al., 2020)*†</td>
<td>51.24</td>
<td>-</td>
</tr>
<tr>
<td>MinTL†</td>
<td>51.00</td>
<td>12.8</td>
</tr>
<tr>
<td>BM25-Retrieval DST†</td>
<td>51.20</td>
<td>12.8</td>
</tr>
<tr>
<td>RoBERTa-Retrieval DST†</td>
<td>51.50</td>
<td>12.7</td>
</tr>
<tr>
<td>NRA-DST†</td>
<td>51.90</td>
<td>12.5</td>
</tr>
</tbody>
</table>

Table 3: Ablation comparing different choices of creating neural index and neural retrieval.

<table>
<thead>
<tr>
<th>Previous Dialogue State</th>
<th>Delexicalised Utterances</th>
<th>Joint Accuracy (top1)</th>
<th>Joint Accuracy (top3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>50.8</td>
<td>50.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>50.8</td>
<td>50.9</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>51.3</td>
<td>51.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td><strong>51.9</strong></td>
<td><strong>51.2</strong></td>
</tr>
</tbody>
</table>

Table 4: Ablation comparing conditioning retrieval result at encoder and decoder.

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Slot Detection Accuracy (↑)</th>
<th>Error (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoder-NRADST</td>
<td>50.9</td>
<td>12.6</td>
</tr>
<tr>
<td>Encoder-NRADST</td>
<td><strong>51.9</strong></td>
<td><strong>12.5</strong></td>
</tr>
</tbody>
</table>

5 Ablation Analysis

We analyze the influence of different changes on the results with the following experiments. We try to analyze the importance of previous dialogue state information and delexicalization while creating the neural index and conditioning retrieved results at the encoder or the decoder of our DST model.

5.1 Neural Index

To understand optimal method for neural index preparation, we investigate the effect of using previous dialogue state and delexicalization. Delexicalization is done on the entire dialogue context $c_t$, which includes removing the slot values from the previous dialogue state and delexicalizing exact slot values from user and system responses. As seen in Table 3, using previous dialogue and delexicalization is very effective.

6 Conclusions

In this work, we demonstrated that neural retrieval augmentation increases the performance of generation-based DST. We explore a simple pipeline resulting in a retrieval-guided generation approach for DST. Moreover, our experiments and ablation studies indicate that neural retrieval can efficiently retrieve a combination of unstructured data (dialogue state) and structured data (user/system utterances). As a result, we improve the performance of the baseline approach on a large-scale multi-domain dataset, MultiWOZ 2.0. In future work, we will investigate the end-to-end training of our NRA-DST framework.
References


