Improving Long Distance Slot Carryover in Spoken Dialogue Systems

Tongfei Chen* **Chetan Naik†** **Hua He†**
Pushpendre Rastogi† **Lambert Mathias†**
* Johns Hopkins University
† Amazon.com, Inc.
tongfei@jhu.edu, {chetnaik,huhe,prastogi,mathiasl}@amazon.com

Abstract

Tracking the state of the conversation is a central component in task-oriented spoken dialogue systems. One such approach for tracking the dialogue state is *slot carryover*, where a model makes a binary decision if a slot from the context is relevant to the current turn. Previous work on the slot carryover task used models that made independent decisions for each slot. A close analysis of the results show that this approach results in poor performance over longer context dialogues. In this paper, we propose to jointly model the slots. We propose two neural network architectures, one based on pointer networks that incorporate slot ordering information, and the other based on transformer networks that uses self attention mechanism to model the slot interdependencies. Our experiments on an internal dialogue benchmark dataset and on the public DSTC2 dataset demonstrate that our proposed models are able to resolve longer distance slot references and are able to achieve competitive performance.

1 Introduction

In task-oriented spoken dialogue systems, the user and the system are engaged in interactions that can span multiple turns. A key challenge here is that the user can reference entities introduced in previous dialogue turns. For example, if a user request for *what's the weather in arlington* is followed by *how about tomorrow*, the dialogue system has to keep track of the entity *arlington* being referenced.

In *slot-based* spoken dialogue systems, tracking the entities in context can be cast as *slot carryover* task – only the relevant slots from the dialogue context are carried over to the current turn. Recent work by Naik et al. (2018) describes a scalable multi-domain neural network architecture to address the task in a diverse schema setting. However, this approach treats every slot as independent. Consequently, as shown in our experiments, this results in lower performance when the contextual slot being referenced is associated with dialogue turns that are further away from the current turn. We posit that modeling slots jointly is essential for improving the accuracy over long distances, particularly when slots are correlated. We motivate this with an example conversation in Figure 1. In this example, the slots WEATHERCITY/WEATHERSTATE, need to be carried over *together* from dialogue history as they are correlated. However, the model in Naik et al. (2018) has no information about this slot interdependence and may choose to carryover only one of the slots. In this work, we alleviate this issue by propos-

Figure 1: An example of a conversation session. Slots are listed on the right. Related slots often co-occur, such as (1) [WEATHERCITY: *San Francisco*] and [WEATHERSTATE: *CA*], and should be carried over together due to their interdependencies (2) PLACE slot is often seen to occur along with TOWN.

Proceedings of the 1st Workshop on NLP for Conversational AI, pages 96–105
Florence, Italy, August 1, 2019.

ing two novel neural network architectures – one based on pointer networks (Vinyals et al., 2015) and another based on self-attention with transformers (Vaswani et al., 2017) – that can learn to *jointly* predict jointly whether a subset of related slots should be carried over from dialogue history.

To validate our approach, we conduct thorough evaluations on both the publicly available DSTC2 task (Henderson et al., 2014), as well as our internal dialogue dataset collected from a commercial digital assistant. In Section 4.3, we show that our proposed approach improve slot carryover accuracy over the baseline systems over longer dialogue contexts. A detailed error analysis reveals that our proposed models are more likely to utilize "anchor" slots – slots tagged in the current utterance – to carry over long-distance slots from context.

To summarize we make the following contributions in this work:

1. We improve upon the slot carryover model architecture in Naik et al. (2018) by introducing approaches for modeling slot interdependencies. We propose two neural network models based on pointer networks and transformer networks that can make joint predictions over slots.
2. We provide a detailed analysis of the proposed models both on an internal benchmark and public dataset. We show that contextual encoding of slots and modeling slot interdependencies is essential for improving performance of slot carryover over longer dialogue contexts. Transformer architectures with self attention provide the best performance overall.

2 Problem Formulation

A dialogue H is formulated as a sequence of utterances, alternatively uttered by a user (U) and the system agent (A):

$$H = \left(h_d^{\{U,A\}}, \cdots, h_2^{\mathrm{U}}, h_1^{\mathrm{A}}, h_0^{\mathrm{U}}\right), \tag{1}$$

where each element h is an utterance. A subscript d denotes the utterance distance which measures the offset from the most recent user utterance (h_0^{U}). The i-th token of an utterance with distance d is denoted as $h_d[i]$.

A slot $x = (d, k, l, r)$ in a dialogue is defined as a key-value pair that contains an entity information, e.g. [CITY:*San Francisco*]. Each slot can be determined by the utterance distance d, slot key k, and a span $[l : r]$ over the tokens of the utterance with slot value represented as $h_d[l : r]$.

Given a dialogue history H and a set of candidate slots X, the context carryover task is addressed by deciding which slots should be carried over. The previous work (Naik et al., 2018) addressed the task as a binary classification problem and each slot $x \subseteq X$ is classified independently. In contrast, our proposed models can explicitly capture slot interactions and make joint predictions of all slots. We show formulations of both model types below,

$$F_{\text{binary}}(x, H) \in (0, 1) \qquad \forall x \in X \tag{2}$$

$$F_{\text{joint}}(X, H) \subseteq X \tag{3}$$

where $F_{\text{binary}}(x, H)$ denotes a binary classification model (Naik et al., 2018), $F_{\text{joint}}(X, H)$ denotes our joint prediction models.

3 Models

3.1 General architecture

Candidate Generation We follow the approach in Naik et al. (2018), where, given a dialogue H, we construct a candidate set of slots X from the context by leveraging the slot key embeddings to find the nearest slot keys that are associated with the current turn.

Slot Encoder A model, given a candidate slot (a slot key, a span in the history and a distance), results in a fixed-length vector representation of a slot: $\mathbf{x} = F_{\mathrm{S}}(x, H) \in \mathbb{R}^{D_{\mathrm{S}}}$, where x is the slot, H is the full history.

Dialogue Encoder We serialize the utterances in the dialogue and use BiLSTM to encode the context as a fixed-length vector $\mathbf{c} = \text{BiLSTM}(H) \in \mathbb{R}^{D_{\mathrm{C}}}$.

Intent Encoder The intent I of the most recent utterance determined by an NLU module is also encoded as a fixed-length vector $\mathbf{i} \in \mathbb{R}^{D_{\mathrm{I}}}$ by averaging the tokens in the intent. We average the word embeddings of the tokens associated with the intent to get the intent embedding.

Decoder Given the encoded vector representations $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ of the slots, the context vector $\mathbf{c}$, the intent vector $\mathbf{i}$, produce a subset of the slot ids:

$$F_{\mathrm{D}}(\mathbf{x}_{1:n}, \mathbf{c}, \mathbf{i}) \subseteq \{1, \cdots, N\} \tag{4}$$

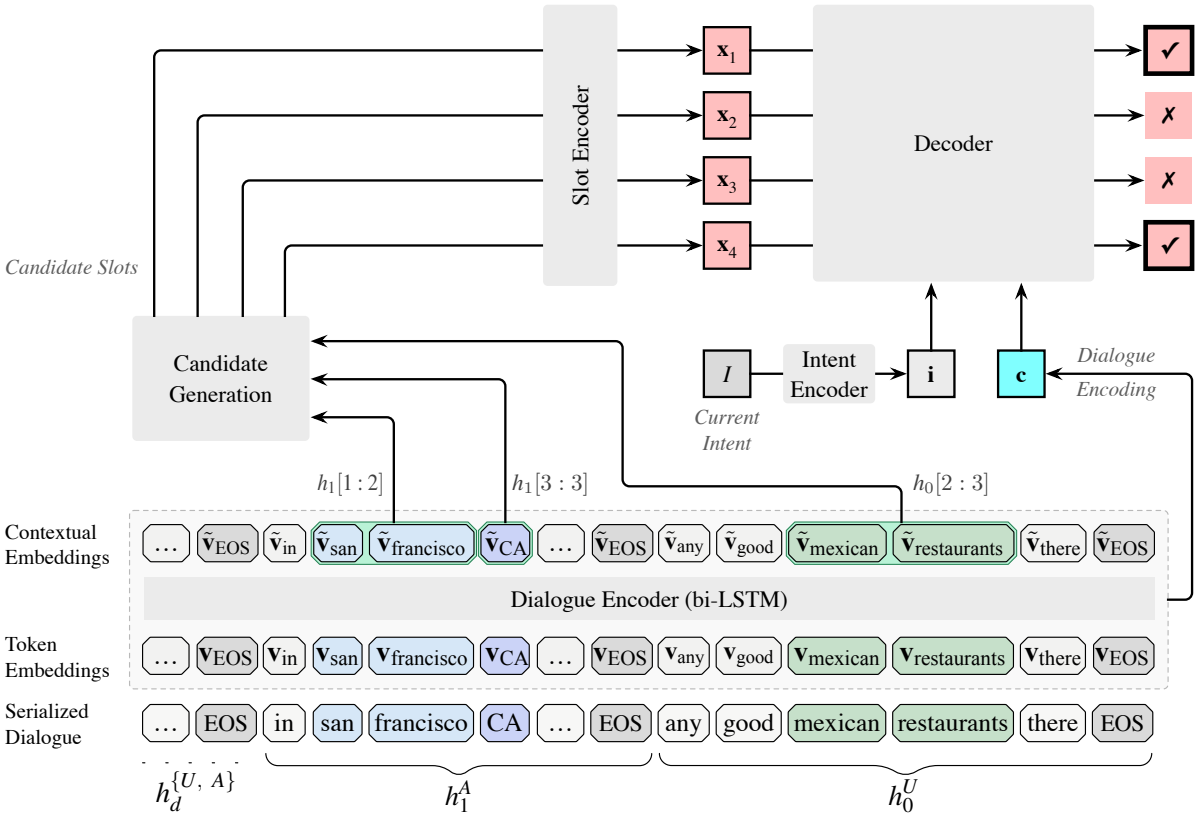


Figure 2: General architecture of the proposed contextual carryover model. Bi-LSTM is used to encode the utterances in dialogue into a fixed length dialogue representation and also get contextual slot value embeddings. Slot encoder uses the slot key, value and distance to create a fixed length slot embedding for each of the candidate slots. Given the encoded slots, intent and dialogue context, decoder selects the subset of slots that are relevant for the current user request.

The overall architecture of the model is shown in Figure 2. We elaborate on the specific designs of these components under this general architecture.

3.2 Slot Encoder Variants

In this section, we describe the different encoding methods that we use to encode slots.

We average the word embeddings of the tokens in the slot key as the *slot key encoding*:

$$\mathbf{x}_{\text{key}} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{v}(k_i)\ . \tag{5}$$

where $\mathbf{v}(w)$ is the embedding vector of token w.

For the slot value (the tokens $h_d[l:r]$), we propose following encoding approaches.

CTX$_{\text{avg}}$ The first is to average the token embeddings of the tokens in the slot value:

$$\mathbf{x}_{\text{val}} = \frac{1}{r-l+1}\sum_{i=l}^{r}\mathbf{v}(h_d[i])\ ; \tag{6}$$

CTX$_{\text{LSTM}}$ To get improved contextualized representation of the slot value in dialogue, we also use neural network models to encode slots. We experimented with bidirectional LSTM (Hochreiter and Schmidhuber, 1997) model for slot encoding. LSTMs are equipped with feedback loops in their recurrent layer, which helps store contextual information over a long history. We encode all dialogue utterances with BiLSTM to obtain contextualized vector representations $\tilde{\mathbf{v}}(w)$ for each token w, then average the output hidden states of the tokens in the span $[l:r]$ to get the slot value encoding.

$$\mathbf{x}_{\text{val}} = \frac{1}{r-l+1}\sum_{i=l}^{r}\tilde{\mathbf{v}}(h_d[i]); \tag{7}$$

Additionally, *distance* may contain important signals. This integer, being odd or even, provides information on whether this utterance is uttered by a user or the system. The smaller it is, the closer a slot is to the current utterance, hence implicitly more probable to be carried over. Building on these intuitions, we encode the distance as a small

vector ($\mathbf{x}_{\text{dist}}$, 4 dimensions) and append it to the overall slot encoding:

$$\mathbf{x} = \left[\mathbf{x}_{\text{key}} \; ; \; \mathbf{x}_{\text{val}} \; ; \; \mathbf{x}_{\text{dist}}\right] . \tag{8}$$

3.3 Decoder Variants

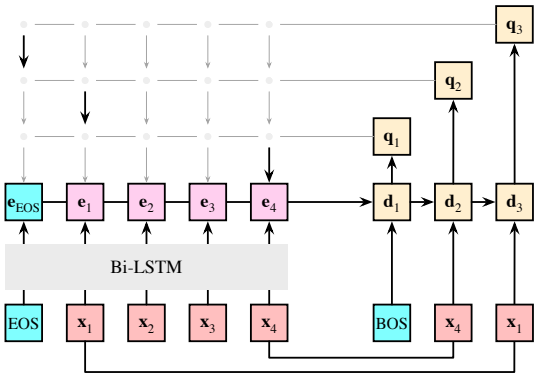


Figure 3: Architecture of the pointer network decoder. In this case, the pointer network selects $\mathbf{x}_4$, $\mathbf{x}_1$ successively and stops after selecting EOS.

Pointer network decoder We adopt the architecture of the pointer network (Vinyals et al., 2015) as a method to perform joint prediction of the slots to be carried over. Pointer networks, a variant of Seq2Seq (Bahdanau et al., 2015; Sutskever et al., 2014; Luong et al., 2015) model, instead of transducing the input sequence into another output sequence, yields a succession of soft pointers (attention vectors) to the input sequence, hence producing an ordering of the elements of a variable-length input sequence.

We use a pointer network to select a *subset* of the slots from the input slot set. The input slot encodings are ordered as a sequence, then fed into a bidirectional LSTM encoder to yield a sequence of encoded hidden states. We experiment with different slot orderings as described in section 4.

$$\mathbf{e}_{0:n} = \text{BiLSTM}([\mathbf{x}_{\text{EOS}} \; , \; \mathbf{x}_{1:n}]) \tag{9}$$

Here a special sentinel token EOS is appended to the beginning of the input to the pointer network – when decoding, once the output pointer points to this EOS token, the decoding process stops.

Given the hidden states, $\mathbf{e}_{0:n}$, the decoding process at every time step i is computed and updated as shown in Algorithm 1.

Contrary to normal attention-based models which directly uses the decoder state ($\mathbf{d}_i$) as the query, we incorporate the context vector ($\mathbf{c}$) and the intent vector ($\mathbf{i}$) into the attention query. The

Algorithm 1 Pointer network decoding

procedure PTRNETDEC($\mathbf{x}_{0:n}, \mathbf{e}_{0:n}, \mathbf{d}_0, \mathbf{c}, \mathbf{i}$)
  $i \leftarrow 0$
  $y_0 \leftarrow$ BOS ▷ special BOS token
  $m_{0:n} \leftarrow$ TRUE ▷ every slot is available
  repeat
    $i \leftarrow i + 1$
    $\mathbf{d}_i \leftarrow \text{LSTM}(\mathbf{d}_{i-1}, \mathbf{x}_{y_{i-1}})$ ▷ update state
    $\mathbf{q}_i \leftarrow F_{\text{Q}}(\mathbf{d}_i, \mathbf{c}, \mathbf{i})$ ▷ constructs query
    $a_{ij} \leftarrow F_{\text{A}}(\mathbf{q}_i, \mathbf{e}_j)$ ▷ attention scores
    $p_{ij} \leftarrow \dfrac{\exp a_{ij}}{\sum_{m_j = \text{TRUE}} \exp a_{ij}}$ ▷ soft pointer
    $\hat{y}_i \leftarrow \arg\max_{m_j = \text{TRUE}} p_{ij}$ ▷ predicted output
    if at inference time **then**
      $y_i \leftarrow \hat{y}_i$ ▷ no gold output
    end if
    $m_{y_i} \leftarrow$ FALSE ▷ update mask
  until $y_i = 0$ ▷ index of EOS is 0
  return $\hat{y}_{1:i-1}$ ▷ return all generated $\hat{y}$'s
end procedure

query vector is a concatenation of the three components:

$$\mathbf{q}_i = F_{\text{Q}}(\mathbf{d}_i, \mathbf{c}, \mathbf{i}) = [\, \mathbf{d}_i \; ; \; \mathbf{c} \; ; \; \mathbf{i} \,] . \tag{10}$$

We use the general Luong attention (Luong et al., 2015) scoring function (bilinear form):

$$a_{ij} = F_{\text{A}}(\mathbf{q}_i, \mathbf{e}_j) = \mathbf{q}_i^{\text{T}} \mathbf{W} \mathbf{e}_j \; . \tag{11}$$

As a subset output is desired, the output $\hat{y}_i$ should be distinct at each step i. To this end, we utilize a *dynamic mask* in the decoding process: for every input slot encoding $\mathbf{x}_j$ a Boolean mask variable m_j is set to TRUE. Once a specific slot is generated, it is crossed out – its corresponding mask is set to FALSE, and further pointers will never attend to this slot again. Hence distinctness of the output sequence is ensured.

Self-attention decoder The pointer network as introduced previously yields a succession of pointers that select slots based on attention scores, which allows the model to look back and forth over entire slot sequence for slot dependency modeling. Similar to the pointer network, the self-attention mechanism is also capable of modeling relationships between all slots in the dialogue, regardless of their respective positions. To compute the representation of any given slot, the self-attention model compares it to every other slot in

the dialogue. The result of these comparisons is attention scores which determine how much each of the other slots should contribute to the representation of the given slot. In this section, we also propose to use the self-attention mechanism with the neural transformer networks (Vaswani et al., 2017) to model slot interdependencies for the task.

One major component in the transformer is the multi-head self-attention unit. Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times and allows the model to jointly attend to information from different perspectives at different positions, which is empirically shown to be more powerful than a single attention head (Vaswani et al., 2017). In our configurations, we increase the number of heads Z, as described in section 4. The independent attention head g outputs are simply concatenated and linearly transformed into the expected output.

Given the input slot encodings $\mathbf{x}_{1:n}$, we compute the self-attention as follows:

$$\mathbf{q}_i^z = \mathbf{W}_{\mathrm{Q}}^z F_{\mathrm{Q}}(\mathbf{x}_i) \tag{12}$$

$$\mathbf{k}_i^z = \mathbf{W}_{\mathrm{K}}^z \mathbf{x}_i \tag{13}$$

$$a_{ij}^z = F_{\mathrm{A}}(\mathbf{q}_i^z, \mathbf{k}_j^z) \tag{14}$$

$$p_{ij}^z = \frac{\exp a_{ij}^z}{\sum_j \exp a_{ij}^z} \tag{15}$$

$$\mathbf{o}_i^z = \sum_j p_{ij}^z \mathbf{k}_j^z \tag{16}$$

$$\tilde{\mathbf{x}}_i = \mathbf{W}_{\mathrm{O}} \left[\mathbf{o}_i^0 \,;\, \cdots \,;\, \mathbf{o}_i^{Z-1}\right] + \mathbf{b}_{\mathrm{O}} \tag{17}$$

where the superscript $0 \leq z < Z$ is the head number. We model the query construction, Equation 12, and the attention score, Equation 14, in the same way as their counterparts (Equation 10 and Equation 11) in the previous pointer network model. The self-attended representation of slot i, $\tilde{\mathbf{x}}_i$, is a representation of slot i with the relations to all other slots taken into account.

We derive the final decision over whether to carry over a slot as a 2-layer feedforward neural network atop the features $\mathbf{x}_i$, $\tilde{\mathbf{x}}_i$, context vector ($\mathbf{c}$) and the intent vector ($\mathbf{i}$):

$$y_i = \sigma(\mathbf{W}_2 \cdot \mathrm{ReLU}(\mathbf{W}_1[\,\mathbf{x}_i \,;\, \tilde{\mathbf{x}}_i \,;\, \mathbf{c} \,;\, \mathbf{i}\,] + \mathbf{b}_1) + \mathbf{b}_2)\,.$$

This creates a highway network connection (Srivastava et al., 2015) that connects the input and the self-attention transformed encodings.

Split		Slot distance			
		0	**1**	**2**	**≥3**
Train	Positive	183K	48K	6.7K	591
	Total	183K	327K	111K	108K
Dev	Positive	22K	6.0K	785	66
	Total	22K	40K	13K	13K
Test	Positive	23K	6.1K	807	85
	Total	23K	41K	13K	14K

Table 1: **Internal Dataset** breakdown showing the number of carryover candidate slots at different distances. 'Total' shows the total number of candidate slots and 'Positive' shows the number of candidate slots that are relevant for the current turn.

Split		Slot distance			
		0	**2**	**4**	**≥6**
Train	Positive	4.6K	3.8K	3.7K	9.6K
	Total	5.2K	4.9K	4.7K	14.5K
Dev	Positive	1.4K	1.2K	1.1K	3.0K
	Total	1.7K	1.6K	1.5K	5.0K
Test	Positive	4.1K	3.2K	3.0K	9.4K
	Total	4.8K	4.2K	3.9K	15.2K

Table 2: **DSTC2 Dataset** breakdown showing the number of carryover candidate slots at different distances. 'Total' shows the total number of candidate slots and 'Positive' shows the number of candidate slots that represent the user goal at the current turn.

4 Experiments

4.1 Datasets

We evaluate our approaches on both internal and external datasets. The internal dataset contains dialogues collected specifically for reference resolution, while the external dataset was collected for dialogue state tracking.

Internal This dataset is made up of a subset of user-initiated dialogue data collected from a commercial voice-based digital assistant. This dataset has 156K dialogues from 7 domains – Music, Q&A, Video, Weather, Local Businesses and Home Automation. Each domain has its own schema. There are ~13 distinct slot keys per domain and only 20% of these keys are reused in more than one domain. To handle dialogue data belonging to a diverse schema, slots in dialogue are converted into candidate slots in the schema associated with the current domain. We follow the

same slot candidate generation recipe by leveraging slot key embedding similarities as in Naik et al. (2018). These candidates are then presented to the models for selecting a subset of relevant candidate slots. Statistics for the candidate slots in the train, development, and test sets broken down by slot distances are shown in Table 1.

DSTC2 The DSTC2 dataset (Henderson et al., 2014) contains system-initiated dialogues between human and dialogue systems in restaurant booking domain. We use top ASR hypothesis as the user utterance and use all the slots from n-best SLU with score > 0.1 as candidate slots. These candidates are then presented to the models for selecting a subset of candidate slots which represent the user goal. Statistics for the candidate slots in the train, development, and test sets broken down by slot distances are shown in Table 2. Since only the user mentioned slots contribute to the user-goal, there are no candidates with odd-numbered slot distances.

4.2 Experimental setup

For all the models, we initialize the word embeddings using fastText embeddings (Lample et al., 2018). The models are trained using mini-batch SGD with Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 to minimize the negative log-likelihood loss. We set the dropout rate of 0.3 for our models during training. In our experiments, we use 300 dimensions for the LSTM hidden states in the pointer network encoder and decoder. Our transformer decoder has 1 layer, $Z = 80$ heads, $d_k = d_v = 64$ for the projection size of keys and values in the attention heads. We do not use positional encoding for the transformer decoder. All pointer network model setups are trained for 40 epochs, our transformer models are trained for 200 epochs. For evaluation on the test set, we pick the best model based on performance on dev set. We use standard definitions of precision, recall, and F1 by comparing the reference slots with the model hypothesis slots.

4.3 Results and discussion

We compare our models against the baseline model – encoder-decoder with word attention architecture described by Naik et al. (2018). Table 3 shows the performance of the models for slots at different distances on Internal dataset.

Impact of slot ordering Using pointer network model, we experiment with the following slot orderings to measure the impact of the order on carryover performance. *no order* – slots are ordered completely randomly. *turn-only order* – slots are ordered based on their slot distance, but the slots with the same distance (i.e., candidates generated from the same contextual turn) are ordered randomly. *temporal order* – slots are ordered based on the order in which they occur in the dialogue.

Partial ordering slots across turns i.e., *turn-only order* significantly improves the carryover performance as compared to using *no order*. Further, enforcing within distance order using *temporal order* improves the overall performance slightly, but we see drop in F1 by 7 points for slots at distance ≥3. indicating that a strict ordering might hurt model accuracy.

Impact of slot encoding Here, we compare slot value representations obtained by averaging pretrained embeddings (CTX_{avg}) with contextualized slot value representation obtained from BiLSTM over complete dialogue(CTX_{LSTM}). The results in Table 3, show that contextualized slot value representation substantially improves model performance compared to the non-contextual representation. This is aligned with the observations on other tasks using contextual word vectors (Peters et al., 2018a; Howard and Ruder, 2018; Devlin et al., 2019).

Impact of decoder Compared to the baseline model, both the pointer network model and the transformer model are able to carry over longer dialogue context due to being able to model the slot interdependence. With the transformer network, we completely forgo ordering information. Though the slot embedding includes distance feature $\mathbf{x}_{\text{dist}}$, the actual order in which the slots are arranged does not matter. We see improvement in carryover performance for slots at all distances. While the pointer network seems to deal with longer context better, the transformer architecture still gives us the best overall performance.

For completeness, Table 4 shows the performance on DSTC2 public dataset, where similar conclusions hold.

4.4 Error Analysis

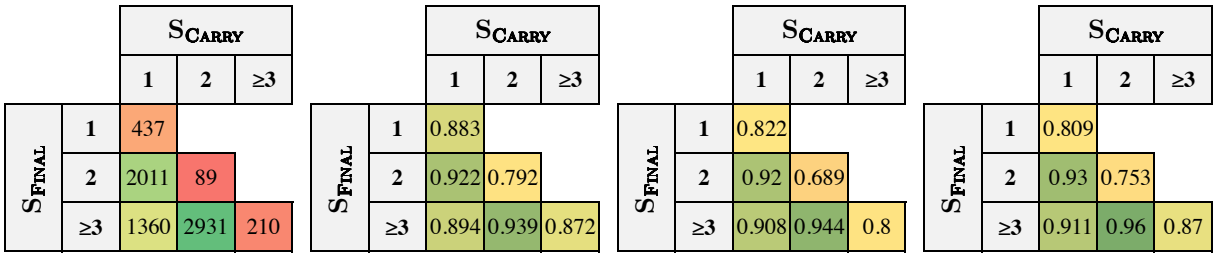
To gain deeper insight into the ability of the models to learn and utilize slot co-occurrence patterns, we measure the models' performance on buckets

Decoder	Slot Encoder	Slot Ordering	Slot distance			
			1	**2**	**≥3**	**≥1**
Baseline (Naik et al., 2018)			**0.8818**	0.6551	0.0000	0.8506
Pointer Network Decoder	CTX_{LSTM}	*no order*	0.8155	0.5571	0.1290	0.7817
	CTX_{LSTM}	*turn-only order*	0.8466	0.6154	**0.4095**	0.8157
	CTX_{avg}	*temporal order*	0.7565	0.4716	0.0225	0.7166
	CTX_{LSTM}	*temporal order*	0.8631	0.6623	0.3350	0.8318
Transformer Decoder	CTX_{LSTM}		0.8771	**0.7035**	0.3803	**0.8533**

Table 3: Carryover performance (F1) of different models for slots at different distances on Internal dataset. The rightmost column contains the aggregate scores for all slots with distance greater than or equal to 1.

Decoder	Slot Encoder	Slot Ordering	Slot distance			
			0	**2**	**4**	**≥6**
Baseline (Naik et al., 2018)			0.9242	0.9111	0.9134	0.8799
Pointer Network Decoder	CTX_{LSTM}	*no order*	0.8316	0.8199	0.8183	0.7641
	CTX_{LSTM}	*turn-only order*	0.9049	0.8993	0.9145	0.8892
	CTX_{LSTM}	*temporal order*	0.9270	0.9204	**0.9290**	**0.9139**
Transformer Decoder	CTX_{LSTM}		**0.9300**	**0.9269**	0.9280	0.8949

Table 4: Carryover performance (F1) of different models for slots at different distances on DSTC2 dataset.



(a) Number of positive instances in the dataset

(b) Baseline model performance

(c) Pointer network performance

(d) Transformer network performance

Figure 4: On internal dataset, plots comparing the performance (F1) of the models across different subsets of candidates separated based on the number of final slots after resolution (y-axis) and the number of slots that are carried over as part of reference resolution (x-axis)

obtained by slicing the data using S_{FINAL} – total number of slots after resolution (i.e after context carryover) and S_{CARRY} – total number of slots carried from context. For example, in a dialogue, if the current turn utterance has 2 slots, and after reference resolution if we carry 3 slots from context, the values for S_{FINAL} and S_{CARRY} would be 5 and 3 respectively. Figure 4 shows the number of instances in each of these buckets and performance of the baseline model, the best pointer network and transformer models on the internal dataset. We notice that the baseline model performs better than the proposed models for instances in the table diagonal ($S_{FINAL} = S_{CARRY}$). These are the instances where the current turn has no slots, and all the necessary slots for the turn have to be carried from historical context. Proposed models perform better in off-diagonal buckets. We hypothesize that the proposed models use anchor slots (slots in current utterance having slot distance 0 which are always positive) and learn slot co-occurrence of candidate slots from context with these anchor slots to improve resolution (i.e., carryover) from longer distances.

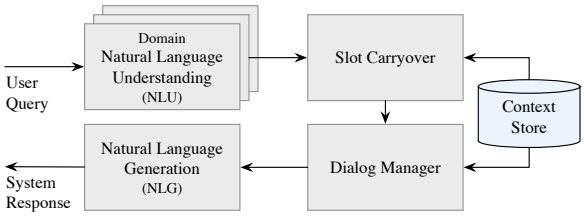


Figure 5: Spoken dialogue system architecture: the reference resolver/context carryover component is used to resolve references in a conversation.

5 Related Work

Figure 5 shows a typical pipelined approach to spoken dialogue (Tur and De Mori, 2011), and where the context carryover system fits into the overall architecture. The context carryover system takes as input, an interpretation output by NLU – typically represented as intents and slots (Wang et al., 2011) – and outputs another interpretation that contains slots from the dialogue context that are relevant to the current turn. The output from context carryover is then fed to the dialogue manager to take the next action. Resolving references to slots in the dialogue plays a vital role in tracking conversation states across turns (Çelikyilmaz et al., 2014). Previous work, e.g., Bhargava et al. (2013); Xu and Sarikaya (2014); Bapna et al. (2017), focus on better leveraging dialogue contexts to improve SLU performance. However, in commercial systems like Siri, Google Assistant, and Alexa, the NLU component is a diverse collection of services spanning rules and statistical models. Typical end-to-end approaches (Bapna et al., 2017) which require back-propagation through the NLU sub-systems are not feasible in this setting.

Dialogue state tracking Dialogue state tracking (DST) focuses on tracking conversational states as well. Traditional DST models rely on hand-crafted semantic delexicalization to achieve generalization (Henderson et al., 2014; Zilka and Jurcícek, 2015; Mrksic et al., 2015). Mrksic et al. (2017) utilize representation learning for states rather than using hand-crafted features. These approaches only operate on fixed ontology and do not generalize well to unknown slot key-value pairs. Rastogi et al. (2017) address this by using sophisticated candidate generation and scoring mechanism while Xu and Hu (2018) use a pointer network to handle unknown slot values. Zhong et al. (2018) share global parameters between estimates for each slot to address extraction of rare slot-value pairs and achieve state-of-the-art on DST. In context carryover, our state tracking does not rely on the definition of user goals and is instead focused on resolving slot references across turns. This approach scales when dealing with multiple spoken language systems, as we do not track the belief states explicitly.

Coreference resolution Our problem is closely related to coreference resolution, where mentions in the current utterance are to be detected and linked to previously mentioned entities. Previous work on coreference resolution have relied on clustering (Bagga and Baldwin, 1998; Stoyanov and Eisner, 2012) or comparing mention pairs (Durrett and Klein, 2013; Wiseman et al., 2015; Sankepally et al., 2018). This has two problems. (1) most traditional methods for coreference resolution follows a pipeline approach, with rich linguistic features, making the system cumbersome and prone to cascading errors; (2) Zero pronouns, intent references and other phenomena in spoken dialogue are hard to capture with this approach (Rao et al., 2015). These problems are circumvented in our approach for slot carryover.

6 Conclusions

In this work, we proposed an improvement to the slot carryover task as defined in Naik et al. (2018). Instead of independent decisions across slots, we proposed two architectures to leverage the slot interdependence – a pointer network architecture and a self-attention and transformer based architecture. Our experiments show that both proposed models are good at carrying over slots over longer dialogue context. The transformer model with its self attention mechanism gives us the best overall performance. Furthermore, our experiments show that temporal ordering of slots in the dialogue matter, since recent slots are more likely to be referred to by users in a spoken dialogue system. Moreover, contextualized encoding of slots is also important, which follows the trend of contextualized embeddings (Peters et al., 2018b).

For future work, we plan to improve these models by encoding the actual dialogue timing information into the contextualized slot embeddings as additional signals. We also plan on exploring the impact of pre-trained representations (Devlin et al., 2019) trained specifically over large-scale dialogues as another way to get improved contextualized slot embeddings.

References

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference.*, pages 79–85.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. 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*.

Ankur Bapna, Gökhan Tür, Dilek Z. Hakkani-Tür, and Larry P. Heck. 2017. Sequential dialogue context modeling for spoken language understanding. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 103–114.

A. Bhargava, Asli Çelikyilmaz, Dilek Hakkani-Tür, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 8337–8341.

Asli Çelikyilmaz, Zhaleh Feizollahi, Dilek Z. Hakkani-Tür, and Ruhi Sarikaya. 2014. Resolving referring expressions in conversational dialogs for natural user interfaces. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 2094–2104.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve J. Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, pages 292–299.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 794–799.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1777–1788.

Chetan Naik, Arpit Gupta, Hancheng Ge, Lambert Mathias, and Ruhi Sarikaya. 2018. Contextual slot carryover for disparate schemas. In *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018.*, pages 596–600.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Associ-*

ation for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers), pages 2227–2237.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sudha Rao, Allyson Ettinger, Hal Daumé III, and Philip Resnik. 2015. Dialogue focus tracking for zero pronoun resolution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 494–503, Denver, Colorado. Association for Computational Linguistics.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry P. Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2017, Okinawa, Japan, December 16-20, 2017*, pages 561–568.

Rashmi Sankepally, Tongfei Chen, Benjamin Van Durme, and Douglas W. Oard. 2018. A test collection for coreferent mention retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1209–1212.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *Computing Research Repository*, arXiv:1505.00387.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 2519–2534.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Gokhan Tur and Renato De Mori. 2011. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. John Wiley and Sons.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Ye-Yi Wang, Li Deng, and Alex Acero. 2011. *Semantic Frame-Based Spoken Language Understanding*, pages 35–80. Wiley.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1416–1426.

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 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1448–1457.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 136–140.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1458–1467.

Lukás Zilka and Filip Jurcícek. 2015. Incremental lstm-based dialog state tracker. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 757–762.