Global-Locally Self-Attentive Dialogue State Tracker

Victor Zhong, Caiming Xiong, Richard Socher
Salesforce Research
Palo Alto, CA
{vzhong, cxiong, rsocher}@salesforce.com

Abstract

Dialogue state tracking, which estimates user goals and requests given the dialogue context, is an essential part of task-oriented dialogue systems. In this paper, we propose the Global-Locally Self-Attentive Dialogue State Tracker (GLAD), which learns representations of the user utterance and previous system actions with global-local modules. Our model uses global modules to share parameters between estimators for different types (called slots) of dialogue states, and uses local modules to learn slot-specific features. We show that this improves tracking of rare states and achieves state-of-the-art performance on the WoZ and DSTC2 state tracking tasks. GLAD obtains 88.1% joint goal accuracy and 97.1% request accuracy on WoZ, outperforming prior work by 3.7% and 5.5%. On DSTC2, our model obtains 74.5% joint goal accuracy and 97.5% request accuracy, outperforming prior work by 1.1% and 1.0%.

1 Introduction

Task oriented dialogue systems can reduce operating costs by automating processes such as call center dispatch and online customer support. Moreover, when combined with automatic speech recognition systems, task-oriented dialogue systems provide the foundation of intelligent assistants such as Amazon Alexa, Apple Siri, and Google Assistant. In turn, these assistants allow for natural, personalized interactions with users by tailoring natural language system responses to the dialogue context. Dialogue state tracking (DST) is a crucial part of dialogue systems (Young et al., 2013). In DST, a dialogue state tracker estimates the state of the conversation using the current user utterance and the conversation history. The dialogue system then uses this estimated state to plan the next action and respond to the user. A state in DST typically consists of a set of *requests* and *joint goals*. Consider the task of restaurant reservation as an example. During each turn, the user informs the system of particular *goals* the they would like to achieve (e.g. `inform(food=french)`), or *request* for more information from the system (e.g. `request(address)`). The set of goal and request slot-value pairs (e.g. `(food, french)`, `(request, address)`) given during a turn are referred to as the *turn goal* and *turn request*. The *joint goal* is the set of accumulated turn goals up to the current turn. Figure 1 shows an example dialogue with annotated turn states, in which the user reserves a restaurant.

Traditional dialogue state trackers rely on Spoken Language Understanding (SLU) systems (Henderson et al., 2012) in order to understand user utterances. These trackers accumulate errors from the SLU, which sometimes do not have the necessary dialogue context to interpret the user utterances. Subsequent DST research forgo the SLU and directly infer the state using the conversation history and the user utterance (Henderson et al., 2014b; Zilka and Jurcicek, 2015; Mrkšić et al., 2015). These trackers rely on hand-crafted semantic dictionaries and delexicalization — the anonymization of slots and values using generic tags — to achieve generalization. Recent work by Mrkšić et al. (2017) apply representation learning using convolutional neural networks to learn features relevant for each state as opposed to hand-crafting features.

A key problem in DST that is not addressed by existing methods is the extraction of rare slot-value pairs that compose the state during each

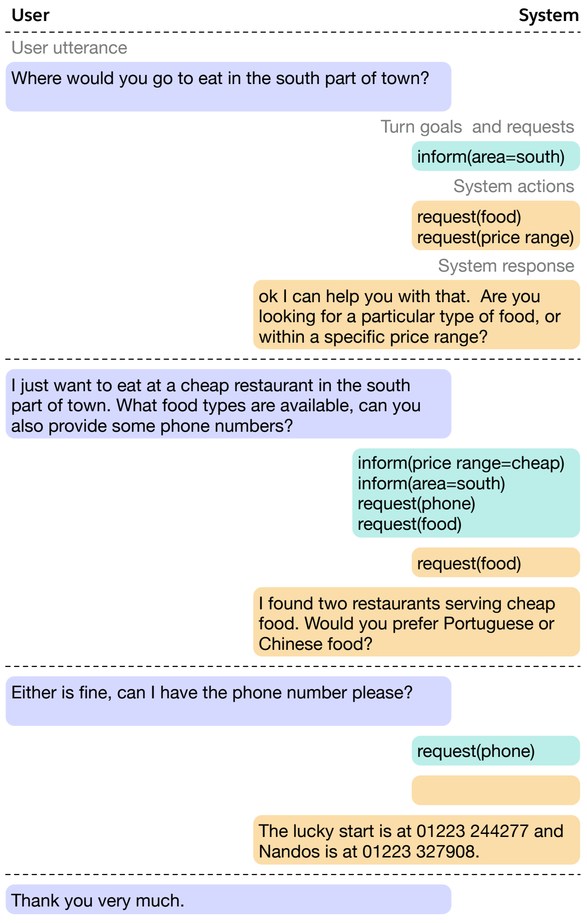
User
System
User utterance
Where would you go to eat in the south part of town?
Turn goals and requests
inform(area=south)
System actions
request(food)
request(price range)
System response
ok I can help you with that. Are you looking for a particular type of food, or within a specific price range?
I just want to eat at a cheap restaurant in the south part of town. What food types are available, can you also provide some phone numbers?
inform(price range=cheap)
inform(area=south)
request(phone)
request(food)
request(food)
I found two restaurants serving cheap food. Would you prefer Portuguese or Chinese food?
Either is fine, can I have the phone number please?
request(phone)
The lucky start is at 01223 244277 and Nandos is at 01223 327908.
Thank you very much.

Figure 1: An example dialogue from the WoZ restaurant reservation corpus. Dashed lines divide turns in the dialogue. A turn contains an user utterance (purple), followed by corresponding turn-level goals and requests (blue). The system then executes actions (yellow), and formulates the result into a natural language response (yellow).

turn. Because task oriented dialogues cover large state spaces, many slot-value pairs that compose the state rarely occur in the training data. Although the chance that the user specifies a particular rare slot-value in a turn is small, the chance that they specify at least one rare slot-value pair is large. Failure to predict these rare slot-value pairs results in incorrect turn-level goal and request tracking. Accumulated errors in turn-level goal tracking significantly degrade joint goal-tracking. For example, in the WoZ state tracking dataset, slot-value pairs have 214.9 training examples on average, while 38.6% of turns have a joint goal that contains a rare slot-value pair with less than 20 training examples.

In this work, we propose the **G**lobal-**L**ocally Self-**A**ttentive **D**ialogue State Tracker (GLAD), a new state-of-the-art model for dialogue state tracking. In contrast to previous work that estimate each slot-value pair independently, GLAD uses global modules to share parameters between estimators for each slot and local modules to learn slot-specific feature representations. We show that by doing so, GLAD generalizes on rare slot-value pairs with few training examples. GLAD achieves state-of-the-art results of 88.1% goal accuracy and 97.1% request accuracy on the WoZ dialogue state tracking task (Wen et al., 2017), outperforming prior best by 3.7% and 5.5%. On DSTC2 (Henderson et al., 2014a), we achieve 74.5% goal accuracy and 97.5% request accuracy, outperforming prior best by 1.1% and 1.0%. We release an implementation of our model along with a Docker image of our experiment setup for reproducibility [1].

2 Global-Locally Self-Attentive Dialogue State Tracker

One formulation of state tracking is to predict the turn state given an user utterance and previous system actions (Williams and Young, 2007). Like previous methods (Henderson et al., 2014b; Wen et al., 2017; Mrkšić et al., 2017), GLAD decomposes the multi-label state prediction problem into a collection of binary prediction problems by using a distinct estimator for each slot-value pair that make up the state. Hence, we describe GLAD with respect to a slot-value pair that is being predicted by the model.

Shown in Figure 2, GLAD is comprised of an encoder module and a scoring module. The encoder module consists of separate global-locally self-attentive encoders for the user utterance, the previous system actions, and the slot-value pair under consideration. The scoring module consists of two scorers. One scorer considers the contribution from the utterance while the other considers the contribution from previous system actions.

2.1 Global-Locally Self-Attentive Encoder

We begin by describing the global-locally self-attentive encoder, which makes up the encoder module. DST datasets tend to be small relative to their state space in that many slot-value pairs rarely occur in the dataset. Because each state is comprised of a set of slot-value pairs, many of

[1]https://github.com/salesforce/glad

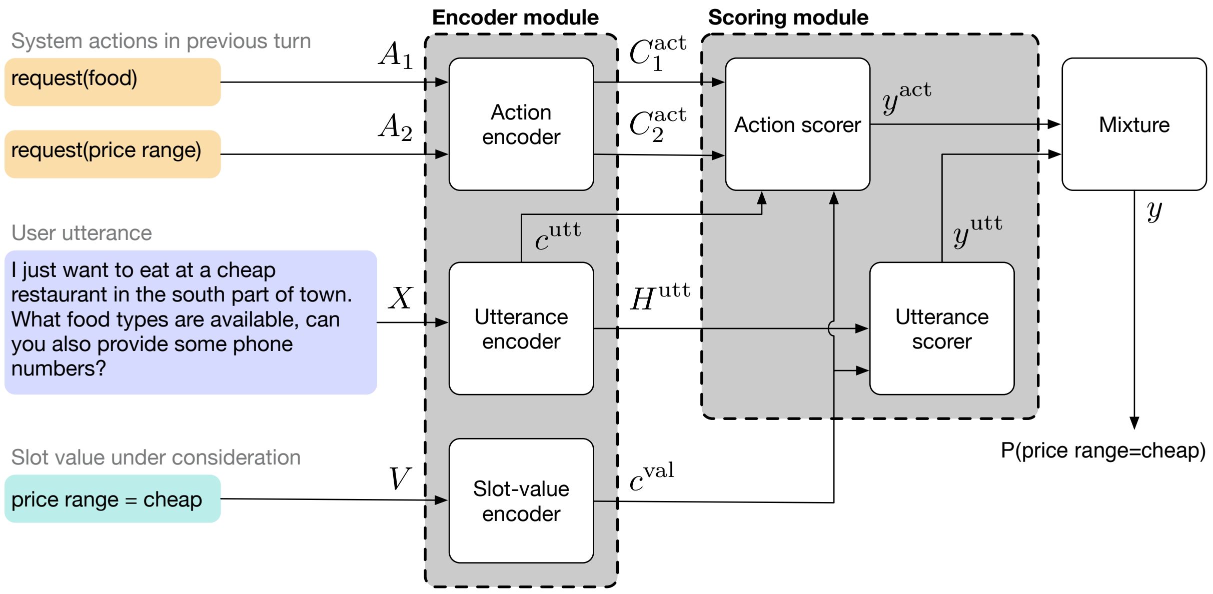

Figure 2: The Global-Locally Self-Attentive Dialogue State Tracker.

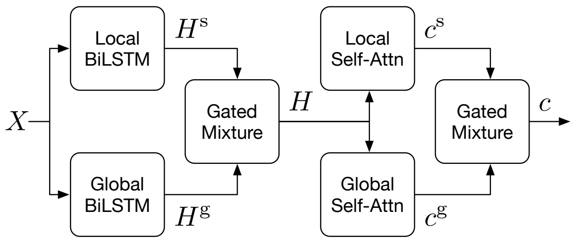

Figure 3: Global-locally self-attentive encoder.

them rare, poor inference of rare slot-value pairs subsequently results in poor turn-level tracking. This problem is amplified in joint tracking, due to the accumulation of turn-level errors. In developing this encoder, we seek to better model rare slot-value pairs by sharing parameters between each slot through global modules and learning slot-specific features through local modules.

The global-locally self-attentive encoder consists of a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), which captures temporal relationships within the sequence, followed by a self-attention layer to compute the summary of the sequence. Figure 3 illustrates the global-locally self-attentive encoder.

Consider the process of encoding a sequence with respect to a particular slot s. Let n denote the number of words in the sequence, d_{emb} the dimension of the embeddings, and $X \in \mathbb{R}^{n \times d_{\text{emb}}}$ the word embeddings corresponding to words in the sequence. We produce a global encoding H^{g} of X using a global bidirectional LSTM.

$$H^{\text{g}} = \text{biLSTM}^{\text{g}}(X) \in \mathbb{R}^{n \times d_{\text{rnn}}} \tag{1}$$

where d_{rnn} is the dimension of the LSTM state. We similarly produce a local encoding H^{s} of X, taking into account the slot s, using a local bidirectional LSTM.

$$H^{\text{s}} = \text{biLSTM}^{\text{s}}(X) \in \mathbb{R}^{n \times d_{\text{rnn}}} \tag{2}$$

The outputs of the two LSTMs are combined through a mixture function to yield a global-local encoding H of X.

$$H = \beta^{\text{s}} H^{\text{s}} + (1 - \beta^{\text{s}}) H^{\text{g}} \in \mathbb{R}^{n \times d_{\text{rnn}}} \tag{3}$$

Here, the scalar β^{s} is a learned parameter between 0 and 1 that is specific to the slot s. Next, we compute a global-local self-attention context c over H. Self-attention, or intra-attention, is an effective method to compute summary context over variable-length sequences for natural language processing tasks (Cheng et al., 2016; Vaswani et al., 2017; He et al., 2017; Lee et al., 2017). In our case, we use a global self-attention module to compute an attention context useful for general-purpose state tracking, as well as a local self-attention module to compute a slot-specific attention context.

For each ith element H_i, we compute a scalar global self-attention score a_i^{g} which is subsequently normalized across all elements using a softmax function.

$$\begin{aligned} a_i^{\mathrm{g}} &= W^{\mathrm{g}} H_i + b^{\mathrm{g}} \in \mathbb{R} && (4) \\ p^{\mathrm{g}} &= \operatorname{softmax}(a^{\mathrm{g}}) \in \mathbb{R}^n && (5) \end{aligned}$$

The global self-attention context c^{g} is then the sum of each element H_i, weighted by the corresponding normalized global self-attention score p_i^{g}.

$$c^{\mathrm{g}} = \sum_i p_i^{\mathrm{g}} H_i \in \mathbb{R}^{d_{\mathrm{rnn}}} \qquad (6)$$

We similarly compute the local self-attention context c^{s}.

$$\begin{aligned} a_i^{\mathrm{s}} &= W^{\mathrm{s}} H_i + b^{\mathrm{s}} \in \mathbb{R} && (7) \\ p^{\mathrm{s}} &= \operatorname{softmax}(a^{\mathrm{s}}) \in \mathbb{R}^n && (8) \\ c^{\mathrm{s}} &= \sum_i p_i^{\mathrm{s}} H_i \in \mathbb{R}^{d_{\mathrm{rnn}}} && (9) \end{aligned}$$

The global-local self-attention context c is the mixture

$$c = \beta^{\mathrm{s}} c^{\mathrm{s}} + (1 - \beta^{\mathrm{s}})\, c^{\mathrm{g}} \in \mathbb{R}^{n \times d_{\mathrm{rnn}}} \qquad (10)$$

For ease of exposition, we define the multi-value encode function $\operatorname{encode}(X)$.

$$\operatorname{encode} : X \to H, c \qquad (11)$$

This function maps the sequence X to the encoding H and the self-attention context c.

2.2 Encoding module

Having defined the global-locally self-attentive encoder, we now build representations for the user utterance, the previous system actions, and the slot-value pair under consideration. Let U denote word embeddings of the user utterance, A_j denote those of the jth previous system action (e.g. `request ( price range )`, and V denote those of the slot-value pair under consideration (e.g. `food = french`). We have

$$\begin{aligned} H^{\mathrm{utt}}, c^{\mathrm{utt}} &= \operatorname{encode}(U) && (12) \\ H_j^{\mathrm{act}}, C_j^{\mathrm{act}} &= \operatorname{encode}(A_j) && (13) \\ H^{\mathrm{val}}, c^{\mathrm{val}} &= \operatorname{encode}(V) && (14) \end{aligned}$$

2.3 Scoring module

Intuitively, we can determine whether the user has expressed the slot-value pair under consideration by examining two input sources. The first source is the user utterance, in which the user directly states the goals and requests. An example of this is the user saying "how about a French restaurant in the centre of town?", after the system asked "how may I help you?" To handle these cases, we determine whether the utterance specifies the slot-value pair. Namely, we attend over the user utterance H^{utt}, taking into account the slot-value pair being considered c^{val}, and use the resulting attention context q^{utt} to score the slot-value pair.

$$\begin{aligned} a_i^{\mathrm{utt}} &= \left(H_i^{\mathrm{utt}}\right)^{\intercal} c^{\mathrm{val}} \in \mathbb{R} && (15) \\ p^{\mathrm{utt}} &= \operatorname{softmax}\left(a^{\mathrm{utt}}\right) \in \mathbb{R}^m && (16) \\ q^{\mathrm{utt}} &= \sum_i p_i^{\mathrm{utt}} H_i^{\mathrm{utt}} \in \mathbb{R}^{d_{\mathrm{rnn}}} && (17) \\ y^{\mathrm{utt}} &= W q^{\mathrm{utt}} + b \in \mathbb{R} && (18) \end{aligned}$$

where m is the number of words in the user utterance. The score y^{utt} indicates the degree to which the value was expressed by the user utterance.

The second source is the previous system actions. This source is informative when the user utterance does not present enough information and instead refers to previous system actions. An example of this is the user saying "yes", after the system asked "would you like a restaurant in the centre of town?" To handle these cases, we examine previous actions after considering the user utterance. First, we attend over the previous action representations $C^{\mathrm{act}} = [C_1^{\mathrm{act}} \cdots C_l^{\mathrm{act}}]$, taking into account the current user utterance c^{utt}. Here, l is the number of previous system actions. Then, we use the similarity between the attention context q^{act} and the slot-value pair c^{val} to score the slot-value pair.

$$\begin{aligned} a_j^{\mathrm{act}} &= \left(C_j^{\mathrm{act}}\right)^{\intercal} c^{\mathrm{utt}} \in \mathbb{R} && (19) \\ p^{\mathrm{act}} &= \operatorname{softmax}\left(a^{\mathrm{act}}\right) \in \mathbb{R}^{l+1} && (20) \\ q^{\mathrm{act}} &= \sum_j p_j^{\mathrm{act}} C_j^{\mathrm{act}} \in \mathbb{R}^{d_{\mathrm{rnn}}} && (21) \\ y^{\mathrm{act}} &= \left(q^{\mathrm{act}}\right)^{\intercal} c^{\mathrm{val}} \in \mathbb{R} && (22) \end{aligned}$$

In addition to real actions, we introduce a sentinel action to each turn which allows the attention mechanism to ignore previous system actions. The score y^{act} indicates the degree to which the value was expressed by the previous actions.

The final score y is then a weighted sum between the two scores y^{utt} and y^{act}, normalized by the sigmoid function σ.

$$y = \sigma\left(y^{\text{utt}} + wy^{\text{act}}\right) \in \mathbb{R} \tag{23}$$

Here, the weight w is a learned parameter.

3 Experiments

3.1 Dataset

The Dialogue Systems Technology Challenges (DSTC) provides a common framework for developing and evaluating dialogue systems and dialogue state trackers (Williams et al., 2013; Henderson et al., 2014a). Under this framework, dialogue semantics such as states and actions are based on a task ontology such as restaurant reservation. During each turn, the user informs the system of particular *goals* (e.g. `inform(food=french)`), or *requests* for more information from the system (e.g. `request(address)`). For instance, `food` and `area` are examples of slots in the DSTC2 task, and `french` and `chinese` are example values within the `food` slot. We train and evaluate our model using DSTC2 as well as the Wizard of Oz (WoZ) restaurant reservation task (Wen et al., 2017), which also adheres to the DSTC framework and has the same ontology as DSTC2.

For DSTC2, it is standard to evaluate using the N-best list of the automatic speech recognition system (ASR) that is included with the dataset. Because of this, each turn in the DSTC2 dataset contains several noisy ASR outputs instead of a noise-free user utterance. The WoZ task does not provide ASR outputs, and we instead train and evaluate using the user utterance.

3.2 Metrics

We evaluate our model using turn-level request tracking accuracy as well as joint goal tracking accuracy. Our definition of GLAD in the previous sections describes how to obtain turn goals and requests. To compute the joint goal, we simply accumulate turn goals. In the event that the current turn goal specifies a slot that has been specified before, the new specification takes precedence. For example, suppose the user specifies a `food=french` restaurant during the current turn. If the joint goal has no existing `food` specifications, then we simply add `food=french` to the joint goal. Alternatively, if `food=thai` had been specified in a previous turn, we simply replace it with `food=french`.

3.3 Implementation Details

We use fixed, pretrained GloVe embeddings (Pennington et al., 2014) as well as character n-gram embeddings (Hashimoto et al., 2017). Each model is trained using ADAM (Kingma and Ba, 2015). For regularization, we apply dropout with 0.2 drop rate (Srivastava et al., 2014) to embeddings and the output of each local and global module. We use the development split for hyperparameter tuning and apply early stopping using the joint goal accuracy.

For the DSTC2 task, we train using transcripts of user utterances and evaluate using the noisy ASR transcriptions. During evaluation, we take the sum of the scores resulting from each ASR output as the output score of a particular slot-value. We then normalize this sum using a sigmoid function as shown in Equation (23). We also apply word dropout, in which the embeddings of a word is randomly set to zero with a probability of 0.3. This accounts for the poor quality of ASR outputs in DSTC2, which frequently miss words in the user utterance. We did not find word dropout to be helpful for the WoZ task, which does not contain noisy ASR outputs.

3.4 Comparison to Existing Methods

Table 1 shows the performance of GLAD compared to previous state-of-the-art models. The delexicalisation models, which replace slots and values in the utterance with generic tags, are from Henderson et al. (2014b) for DSTC2 and Wen et al. (2017) for WoZ. Semantic dictionaries map slot-value pairs to hand-engineered synonyms and phrases. The NBT (Mrkšić et al., 2017) applies CNN over word embeddings learned over a paraphrase database (Wieting et al., 2015) instead of delexicalised n-gram features.

On the WoZ dataset, we find that GLAD significantly improves upon previous state-of-the-art performance by 3.7% on joint goal tracking accuracy and 5.5% on turn-level request tracking accuracy. On the DSTC dataset, which evaluates using noisy ASR outputs instead of user utterances, GLAD improves upon previous state of the art per-

Model	DSTC2		WoZ	
	Joint goal	Turn request	Joint goal	Turn request
Delexicalisation-Based Model	69.1%	95.7%	70.8%	87.1%
Delex. Model + Semantic Dictionary	72.9%	95.7%	83.7%	87.6%
Neural Belief Tracker (NBT) - DNN	72.6%	96.4%	84.4%	91.2%
Neural Belief Tracker (NBT) - CNN	73.4%	96.5%	84.2%	91.6%
GLAD	**74.5± 0.2%**	**97.5± 0.1%**	**88.1± 0.4%**	**97.1± 0.2%**

Table 1: Test accuracies on the DSTC2 and WoZ restaurant reservation datasets. The other models are: delexicalisation DSTC2 (Henderson et al., 2014b), delexicalisation WoZ (Wen et al., 2017), and NBT (Mrkšić et al., 2017). We run 10 models using random seeds with early stopping on the development set, and report the mean and standard deviation test accuracies for each dataset.

Model	Tn goal	Jnt goal	Tn request
GLAD	93.7%	88.8%	97.3%
- global	88.8%	73.4%	97.3%
- local	93.1%	86.6%	95.1%
- self-attn	91.6%	84.4%	97.1%
- LSTM	88.7%	71.5%	93.2%

Table 2: Ablation study showing turn goal, joint goal, and turn request accuracies on the dev. split of the WoZ dataset. For "- self-attn", we use mean-pooling instead of self-attention. For "- LSTM", we compute self-attention over word embeddings.

formance by 1.1% on joint goal tracking accuracy and 1.0% on turn-level request tracking accuracy.

3.5 Ablation study

We perform ablation experiments on the development set to analyze the effectiveness of different components of GLAD. The results of these experiments are shown in Table 2. We also show turn goal accuracy in addition to joint goal accuracy and turn request accuracy for reference.

Temporal order is important for state tracking. We experiment with an embedding-matching variant of GLAD with self-attention but without LSTMs. The weaker performance by this model suggests that representations that capture temporal dependencies is helpful for understanding phrases for state tracking.

Self-attention allows slot-specific, robust feature learning. We observe a consistent drop in performance for models that use mean-pooling over the temporal dimension as opposed to self-attention (Equations (4) to (6)). This stems from the flexibility in the attention context computation afforded by the self-attention mechanism, which allows the model to focus on select words relevant to the slot-value pair under consideration. Figure 4 illustrates an example in which local self-attention modules focus on relevant parts of the utterance. The model attends to relevant phrases that n-gram and embedding matching techniques do not capture (e.g. "within 5 miles" for the "area" slot).

Global-local sharing improves goal tracking. We study the two extremes of sharing between the global module and the local module. The first uses only the local module and results in degradation in goal tracking but does not affect request tracking (e.g. $\beta^s = 1$). This is because the former is a joint prediction over several slot-values with few training examples, whereas the latter predicts a single slot that has the most training examples.

The second uses only the global module and underperforms in both goal tracking and request tracking (e.g. $\beta^s = 0$). This model is less expressive, as it lacks slot-specific specializations except for the final scoring modules.

Figure 5 shows the performance of GLAD and the two sharing variants across different numbers of occurrences in the training data. GLAD consistently outperforms both variants for rare slot-value pairs. For slot-value pairs with an abundance of training data, there is no significant performance difference between models as there is sufficient data to generalize.

3.6 Qualitative analysis

Table 3 shows example predictions by GLAD. In the first example, the user explicitly outlines requests and goals in a single utterance. In the second example, the model previously prompted the user for confirmation of two requests (e.g. for the restaurant's address and phone number), and the user simply answers in the affirmative. In this

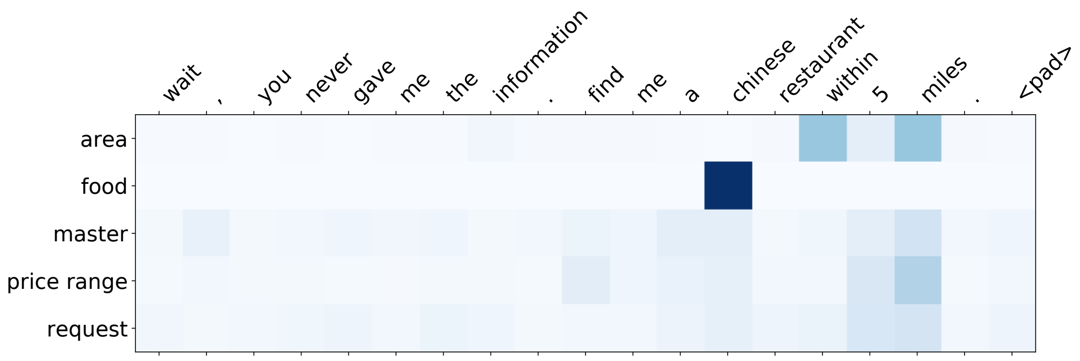

Figure 4: Global and local self-attention scores on user utterances. Each row corresponds to the self-attention score for a particular slot. Slot-specific local self-attention modules emphasize relevant key words and phrases to that slot, whereas the global module attends to all relevant words.

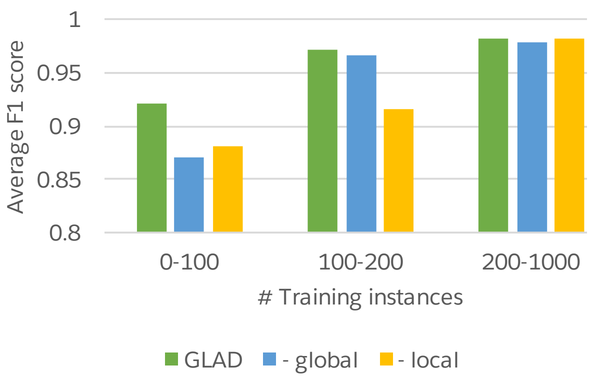

Figure 5: F1 performance for each slot-value pair in the development split of the WoZ task, grouped by the number of training instances.

case, the model obtains the correct result by leveraging the system actions in the previous turn. The last example demonstrates an error made by the model. Here, the user does not answer the system's previous request for the choice of food and instead asks for what food is available. The model misinterprets the lack of response as the user not caring about the choice of food.

4 Related Work

Dialogue State Tracking. Traditional dialogue state trackers rely on a separate SLU component that serves as the initial stage in the dialogue agent pipeline. The downstream tracker then combines the semantics extracted by the SLU with previous dialogue context in order to estimate the current dialogue state (Thomson and Young, 2010; Wang and Lemon, 2013; Williams, 2014; Perez and Liu, 2017). Recent results in dialogue state tracking show that it is beneficial to jointly learn speech understanding and dialogue tracking (Henderson et al., 2014b; Zilka and Jurcicek, 2015; Wen et al., 2017). These approaches directly take as input the N-best list produced by the ASR system. By avoiding the accumulation of errors from the initial SLU component, these joint approaches typically achieved stronger performance on tasks such as DSTC2. One drawback to these approaches is that they rely on hand-crafted features and complex domain-specific lexicon (in addition to the ontology), and consequently are difficult to extend and scale to new domains. The recent Neural Belief Tracker (NBT) by Mrkšić et al. (2017) avoids reliance on hand-crafted features and lexicon by using representation learning. The NBT employs convolutional filters over word embeddings in lieu of previously-used hand-engineered features. Moreover, to outperform previous methods, the NBT uses pretrained embeddings tailored to retain semantic relationships by injecting semantic similarity constraints from the Paraphrase Database (Wieting et al., 2015; Ganitkevitch et al., 2013). On the one hand, these specialized embeddings are more difficult to obtain than word embeddings from language modeling. On the other hand, these embeddings are not specific to any dialogue domain and generalize to new domains.

Neural attention models in NLP. Attention mechanisms have led to improvements on a variety of natural language processing tasks. Bahdanau et al. (2015) propose attentional sequence to sequence models for neural machine translation. Luong et al. (2015) analyze various attention techniques and highlight the effectiveness of the simple, parameterless dot product attention. Similar models have also proven successful in tasks such as summarization (See et al., 2017; Paulus et al.,

System actions in previous turn	User utterance	Predicted turn belief state

N/A	I would like Polynesian food in the South part of town. Please send me phone number and address.	request(phone)
		request(address)
		inform(food=polynesian)
		inform(area=south)

request(address)
request(phone)
 	
There is a moderately priced italian place called Pizza hut at cherry hilton. would you like the address and phone number?
	Yes please.	request(phone)
		request(address)

request(food)
request(price range)
 	
ok I can help you with that. Are you looking for a particular type of food, or within a specific price range?
	I just want to eat at a cheap restaurant in the south part of town. What food types are available, can you also provide some phone numbers?	request(phone)
		inform(price range=cheap)
		inform(area=south)
		-inform(food=dontcare)
		+request(food)

Table 3: Example predictions by Global-Locally Self-Attentive Dialogue State Tracker on the development split of the WoZ restaurant reservation dataset. Model predicted slot-value pairs that are not in the ground truth (e.g. false positives) are prefaced with a "+" symbol. Ground truth slot-value pairs that are not predicted by the model (e.g. false negatives) are prefaced with a "-" symbol.

2018). Self-attention, or intra-attention, has led improvements in language modeling, sentiment analysis, natural language inference (Cheng et al., 2016), semantic role labeling (He et al., 2017), and coreference resolution (Lee et al., 2017). Deep self-attention has also achieved state-of-the-art results in machine translation (Vaswani et al., 2017). Coattention, or bidirectional attention that codependently encode two sequences, have led to significant gains in question answering (Xiong et al., 2017; Seo et al., 2017; Xiong et al., 2018) as well as visual question answering (Lu et al., 2016).

Parameter sharing between related tasks. Sharing parameters between related tasks to improve joint performance is prominent in multi-task learning (Caruana, 1998; Thrun, 1996). Early works in multi-tasking use Gaussian processes whose covariance matrix is induced from shared kernels (Lawrence and Platt, 2004; Yu et al., 2005; Seeger et al., 2005; Bonilla et al., 2008). Hashimoto et al. (2017) propose a progressively trained joint model for NLP tasks. When a new task is introduced, a new section is added to the network whose inputs are intermediate representations from sections for previous tasks. In this sense, tasks share parameters in a hierarchical manner. Johnson et al. (2016) propose a single model that jointly learns to translate between multiple language pairs, including one-to-many, many-to-one, and many-to-many translation. Kaiser et al. (2017) propose a model that jointly learns multiple tasks across modalities. Each modality-specific feature extractor extracts a representation that is fed into a shared encoder.

5 Conclusions

We introduced the Global-Locally Self-Attentive Dialogue State Tracker (GLAD), a new state-of-the-art model for dialogue state tracking. At the core of GLAD is the global-locally self-attention encoder, whose global modules allow parameter sharing between slots and local modules allow slot-specific feature learning. This allows GLAD to generalize on rare slot-value pairs with few training data. GLAD achieves state-of-the-art results of 88.1% goal accuracy and 97.1% request accuracy on the WoZ dialogue state tracking task, as well as 74.5% goal accuracy and 97.5% request accuracy on DSTC2.

Acknowledgement

We thank Nikola Mrkšić for helpful discussion and for providing a preprocessed version of the DSTC2 dataset.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Edwin V Bonilla, Kian M Chai, and Christopher Williams. 2008. Multi-task gaussian process prediction. In *NIPS*.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *ACL*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *HLT-NAACL*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *ACL*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *ACL*.

Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative spoken language understanding using word confusion networks. In *Spoken Language Technology Workshop (SLT)*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *SIGDIAL*.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Compututation* 9(8).

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. Technical report, Google.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137* .

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Neil D Lawrence and John C Platt. 2004. Learning to learn with the informative vector machine. In *ICML*.

Kenton Lee, Luheng He, Mike Lewis, and Luke S. Zettlemoyer. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *ACL*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *ACL*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *EACL*.

Abigail See, Peter Liu, and Christopher Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Matthias Seeger, Yee-Whye Teh, and Michael Jordan. 2005. Semiparametric latent factor models. In *AISTATS*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1).

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language* 24(4).

Sebastian Thrun. 1996. Is learning the n-th thing any easier than learning the first? In *NIPS*.

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 *NIPS*.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. In *ACL*.

Jason D Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *SIGDIAL*.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *SIGDIAL*.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language* 21.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5).

Kai Yu, Volker Tresp, and Anton Schwaighofer. 2005. Learning gaussian processes from multiple tasks. In *ICML*.

Lukas Zilka and Filip Jurcicek. 2015. Incremental LSTM-based dialog state tracker. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*.