Multi-task learning for Joint Language Understanding and Dialogue State Tracking

Abhinav Rastogi
Google AI
Mountain View
`abhirast@google.com`

Raghav Gupta
Google AI
Mountain View
`raghavgupta@google.com`

Dilek Hakkani-Tur
Google AI
Mountain View
`dilek@ieee.org`

Abstract

This paper presents a novel approach for multi-task learning of language understanding (LU) and dialogue state tracking (DST) in task-oriented dialogue systems. Multi-task training enables the sharing of the neural network layers responsible for encoding the user utterance for both LU and DST and improves performance while reducing the number of network parameters. In our proposed framework, DST operates on a set of candidate values for each slot that has been mentioned so far. These candidate sets are generated using LU slot annotations for the current user utterance, dialogue acts corresponding to the preceding system utterance and the dialogue state estimated for the previous turn, enabling DST to handle slots with a large or unbounded set of possible values and deal with slot values not seen during training. Furthermore, to bridge the gap between training and inference, we investigate the use of scheduled sampling on LU output for the current user utterance as well as the DST output for the preceding turn.

1 Introduction

Task-oriented dialogue systems interact with users in natural language to accomplish tasks they have in mind, by providing a natural language interface to a backend (API, database or service). State of the art approaches to task-oriented dialogue systems typically consist of a language understanding (LU) component, which estimates the semantic parse of each user utterance and a dialogue state tracking (DST) or belief tracking component, which keeps track of the conversation context and the dialogue state (DS). Typically, DST uses the semantic parse generated by LU to update the DS at every dialogue turn. The DS accumulates the preferences specified by the user over the dialogue and is used to make requests to a backend. The results from the backend and the dialogue state are then used by a dialogue policy module to generate the next system response.

System:	Hello! How can I help?
Acts:	*greeting*
User:	Hello, book me a table for two at Cascal.
Intent:	RESERVE_RESTAURANT
Acts:	*greeting*, *inform*(#people), *inform*(restaurant)
State:	`restaurant=Cascal,#people=two`
System:	I found a table for two at Cascal at 6 pm. Does that work?
Acts:	*offer*(time=6 pm)
User:	6 pm isn't good for us. How about 7 pm?
Acts:	*negate*(time), *inform*(time)
State:	`restaurant=Cascal,#people=two, time=7 pm`

Figure 1: A dialogue with user intent, user and system dialogue acts, and dialogue state.

Pipelining dialogue system components often leads to error propagation, hence joint modeling of these components has recently gained popularity (Henderson et al., 2014; Mrkšić et al., 2017; Liu and Lane, 2017), owing to computational efficiency as well as the potential ability to recover from errors introduced by LU. However, combining joint modeling with the ability to scale to multiple domains and handle slots with a large set of possible values, potentially containing entities not seen during training, are active areas of research.

In this work, we propose a single, joint model for LU and DST trained with multi-task learning. Similar to Liu and Lane 2017, our model employs a hierarchical recurrent neural network to encode the dialogue context. Intermediate feature representations from this network are used for identifying the intent and dialogue acts, and tagging slots

Utterance:	Table	for	two	at	Olive	Garden
	↓	↓	↓	↓	↓	↓
Slot Tags:	O	O	B-#	O	B-rest	I-rest

Figure 2: IOB slot tags for a user utterance. Slot values *#* = *two* and *rest* = *Olive Garden* are obtained from corresponding B and I tags.

in the user utterance. Slot values obtained using these slot tags (as shown in Figure 2) are then used to update the set of candidate values for each slot. Similar to Rastogi et al. 2017, these candidate values are then scored by a recurrent scoring network which is shared across all slots, thus giving an efficient model for DST which can handle new entities that are not present in the training set - i.e., out-of-vocabulary (OOV) slot values.

During inference, the model uses its own predicted slot tags and previous turn dialogue state. However, ground truth slot tags and dialogue state are used for training to ensure stability. Aiming to bridge this gap between training and inference, we also propose a novel scheduled sampling (Bengio et al., 2015) approach to joint language understanding and dialogue state tracking.

The paper is organized as follows: Section 2 presents related work, followed by Section 3 describing the architecture of the dialogue encoder, which encodes the dialogue turns to be used as features by different tasks in our framework. The section also defines and outlines the implementation of the LU and DST tasks. Section 4 describes our setup for scheduled sampling. We then conclude with experiments and discussion of results.

2 Related Work

The initial motivation for dialogue state tracking came from the uncertainty in speech recognition and other sources (Williams and Young, 2007), as well as to provide a comprehensive input to a downstream dialogue policy component deciding the next system action. Proposed belief tracking models have ranged from rule-based (Wang and Lemon, 2013), to generative (Thomson and Young, 2010), discriminative (Henderson et al., 2014), other maximum entropy models (Williams, 2013) and web-style ranking (Williams, 2014).

Language understanding has commonly been modeled as a combination of intent and dialogue act classification and slot tagging (Tur and De Mori, 2011). Recently, recurrent neural network (RNN) based approaches have shown good results for LU. Hakkani-Tür et al. 2016 used a joint RNN for intents, acts and slots to achieve better overall frame accuracy. In addition, models such as Chen et al. 2016, Bapna et al. 2017 and Su et al. 2018 further improve LU results by incorporating context from dialogue history.

Henderson et al. 2014 proposed a single joint model for single-turn LU and multi-turn DST to improve belief tracking performance. However, it relied on manually constructed semantic dictionaries to identify alternative mentions of ontology items that vary lexically or morphologically. Such an approach is not scalable to more complex domains (Mrkšić et al., 2017) as it is challenging to construct semantic dictionaries that can cover all possible entity mentions that occur naturally in a variety of forms in natural language. Mrkšić et al. 2017 proposed the NBT model which eliminates the LU step by directly operating on the user utterance. However, their approach requires iterating through the set of all possible values for a slot, which could be large or potentially unbounded (e.g. date, time, usernames). Perez and Liu 2017 incorporated end-to-end memory networks, as introduced in Sukhbaatar et al. 2015, into state tracking and Liu and Lane 2017 proposed an end-to-end model for belief tracking. However, these two approaches cannot accommodate OOV slot values as they represent DS as a distribution over all possible slot values seen in the training set.

To handle large value sets and OOV slot values, Rastogi et al. 2017 proposed an approach, where a set of value candidates is formed at each turn using dialogue context. The DST then operates on this set of candidates. In this work, we adopt a similar approach, but our focus is on joint modeling of LU and DST, and sampling methods for training them jointly.

3 Model Architecture

Let a dialogue be a sequence of T turns, each turn containing a user utterance and the preceding system dialogue acts output by the dialogue manager. Figure 3 gives an overview of our model architecture, which includes a user utterance encoder, a system act encoder, a state encoder, a slot tagger and a candidate scorer. At each turn $t \in \{1, ..., T\}$, the model takes a dialogue turn and the previous dialogue state D^{t-1} as input and outputs the predicted user intent, user dialogue acts, slot values in the user utterance and the updated

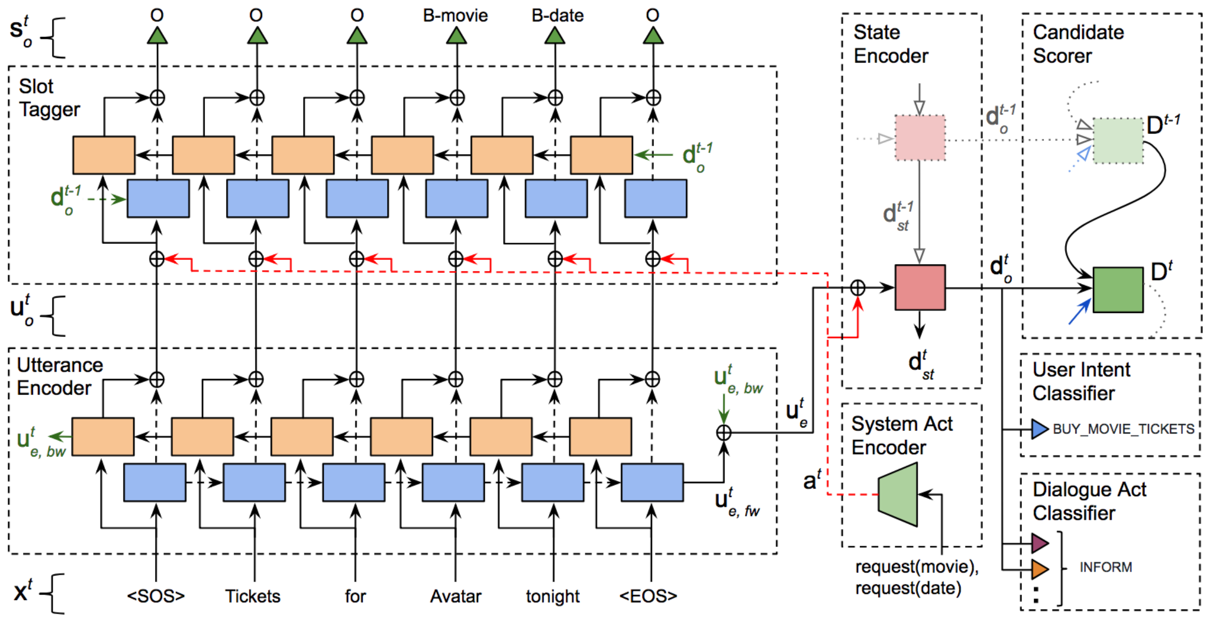


Figure 3: Architecture of our joint LU and DST model as described in Section 3. x^t is the sequence of user utterance token embeddings, a^t is the system act encoding and blue arrows indicate additional features used by DST as detailed in Section 3.8.

dialogue state D^t.

As a new turn arrives, the system act encoder (Section 3.1) encodes all system dialogue acts in the turn to generate the system dialogue act vector a^t. Similarly, the utterance encoder (Section 3.2) encodes the user utterance into a vector u_e^t, and also generates contextual token embeddings u_o^t for each utterance token. The state encoder (Section 3.3) then uses a^t, u_e^t and its previous turn hidden state, d_{st}^{t-1}, to generate the dialogue context vector d_o^t, which summarizes the entire observed dialogue, and its updated hidden state d_{st}^t.

The dialogue context vector d_o^t is then used by the user intent classifier (Section 3.4) and user dialogue act classifier (Section 3.5). The slot tagger (section 3.6) uses the dialogue context from previous turn d_o^{t-1}, the system act vector a^t and contextual token embeddings u_o^t to generate refined contextual token embeddings s_o^t. These refined token embeddings are then used to predict the slot tag for each token in the user utterance.

The system dialogue acts and predicted slot tags are then used to update the set of candidate values for each slot (Section 3.7). The candidate scorer (Section 3.8) then uses the previous dialogue state D^{t-1}, the dialogue context vector d_o^t and other features extracted from the current turn (indicated by blue arrows in Figure 3) to update the scores for all candidates in the candidate set and outputs the updated dialogue state D^t. The following sections describe these components in detail.

3.1 System Act Encoder

Previous turn system dialogue acts play an important role in accurate semantic parsing of a user utterance. Each system dialogue act contains an act type and optional slot and value parameters. The dialogue acts are first encoded into binary vectors denoting the presence of an act type. All dialogue acts which don't have any associated parameters (e.g. *greeting* and *negate*) are encoded as a binary indicator vector a_{utt}^t. Dialogue acts with just a slot s as parameter (e.g. *request(date)*) are encoded as $a_{slot}^t(s)$, whereas acts having a candidate value c for a slot s as parameter (e.g. *offer(time=7pm)*) are encoded as $a_{cand}^t(s,c)$. These binary vectors are then combined using equations 1-4 to obtain the combined system act representation a^t, which is used by other units of dialogue encoder (as shown in Figure 3). In these equations, e_s is a trainable slot embedding defined for each slot s.

$$a_{sc}^t(s) = a_{slot}^t(s) \oplus e_s \oplus \Sigma_c a_{cand}^t(s,c) \tag{1}$$

$$a'^t_{sc}(s) = ReLU(W_{sc}^a \cdot a_{sc}^t(s) + b_{sc}^a) \tag{2}$$

$$a_{usc}^t = \left(\frac{1}{|S^t|}\sum_{s\in S^t} a'^t_{sc}(s)\right) \oplus a_{utt}^t \tag{3}$$

$$a^t = ReLU(W_{usc}^a \cdot a_{usc}^t + b_{usc}^a) \tag{4}$$

3.2 Utterance Encoder

The user utterance takes the tokens corresponding to the user utterance as input. Special tokens `SOS` and `EOS` are added at the beginning and end of the token list. Let $x^t = \{x^t_m \in \mathbb{R}^{u_d}, \forall\, 0 \leq m < M^t\}$ denote the embedded representations of these tokens, where M^t is the number of tokens in the user utterance for turn t (including `SOS` and `EOS`).

We use a single layer bi-directional GRU recurrent neural network (Cho et al., 2014) with state size d_u and initial state set to 0, to encode the user utterance. The first output of the user utterance encoder is $u^t_e \in \mathbb{R}^{2d_u}$, which is a compact representation of the entire user utterance, defined as the concatenation of the final states of the two RNNs. The second output is $u^t_o = \{u^t_{o,m} \in \mathbb{R}^{2d_u}, 0 \leq m < M^t\}$, which is the embedded representation of each token conditioned on the entire utterance, defined as the concatenation of outputs at each step of the forward and backward RNNs.

3.3 State Encoder

The state encoder completes our hierarchical dialogue encoder. At turn t, the state encoder generates d^t_o, which is an embedded representation of the dialogue context until and including turn t. We implement the state encoder using a unidirectional GRU RNN with each timestep corresponding to a dialogue turn. As shown in Figure 3, the dialogue encoder takes $a^t \oplus u^t_e$ and its previous hidden state d^{t-1}_{st} as input and outputs the updated hidden state d^t_{st} and the encoded representation of the dialogue context d^t_o (which are the same in case of GRU).

3.4 User Intent Classification

The user intent is used to identify the backend with which the dialogue system should interact. We predict the intents at each turn to allow user to switch intents during the dialogue. However, we assume that a given user utterance can contain at-most one intent and model intent prediction as a multi-class classification problem. At each turn, the distribution over all intents is calculated as

$$p^t_i = softmax(W_i \cdot d^t_o + b_i) \tag{5}$$

where $\dim(p^t_i) = |I|$, $W_i \in \mathbb{R}^{d \times |I|}$ and $b_i \in \mathbb{R}^{|I|}$, I denoting the user intent vocabulary and $d = \dim(d^t_o)$. During inference, we predict $argmax(p^t_i)$ as the intent label for the utterance.

3.5 User Dialogue Act Classification

Dialogue acts are structured semantic representations of user utterances. User dialogue acts are used by the dialogue manager in deciding the next system action. We model user dialogue act classification as a multilabel classification problem, to allow for the presence of more than one dialogue act in a turn (Tur and De Mori, 2011). At each turn, the probability for act a is predicted as

$$p^t_a = sigmoid(W_a \cdot d^t_o + b_a) \tag{6}$$

where $\dim(p^t_a) = |A_u|$, $W_a \in \mathbb{R}^{d \times |A_u|}$, $b_a \in \mathbb{R}^{|A_u|}$, A_u is the user dialogue act vocabulary and $d = \dim(d^t_o)$. For each act α, $p^t_a(\alpha)$ is interpreted as the probability of presence of α in turn t. During inference, all dialogue acts with a probability greater than t_u are predicted, where $0 < t_u < 1.0$ is a hyperparameter tuned using the dev set.

3.6 Slot Tagging

Slot tagging is the task of identifying the presence of values of different slots in the user utterance. We use the IOB tagging scheme (Tjong Kim Sang and Buchholz 2000, see Figure 2) to assign a label to each token. These labels are then used to extract the values for different slots from the utterance.

The slot tagging network consists of a single-layer bidirectional LSTM RNN (Hochreiter and Schmidhuber, 1997), which takes the contextual token embeddings u^t_o generated by the utterance encoder as input. It outputs refined token embeddings $s^t_o = \{s^t_{o,m}, \forall\, 0 \leq m < M^t\}$ for each token, M^t being the number of tokens in user utterance at turn t.

Models making use of dialogue context for LU have been shown to achieve superior performance (Chen et al., 2016). In our setup, the dialogue context vector d^{t-1}_o encodes all the preceding turns and the system act vector a^t encodes the system dialogue acts preceding the user utterance. As shown in Figure 3, d^{t-1}_o is used to initialize [1] the hidden state (cell states are initialized to zero) for the forward and backward LSTM recurrent units in the slot tagger, while a^t is fed as input to the tagger by concatenating with each element of u^t_o as shown below. We use an LSTM instead of a GRU for this layer since that resulted in better performance on the validation set.

$$s^t_{in} = \{u^t_{o,m} \oplus a^t, \forall\, 0 \leq m < M^t\} \tag{7}$$

$$s^t_{e,bw}, s^t_{o,bw} = LSTM_{bw}(s^t_{in}) \tag{8}$$

$$s^t_{e,fw}, s^t_{o,fw} = LSTM_{fw}(s^t_{in}) \tag{9}$$

$$s^t_o = s^t_{o,bw} \oplus s^t_{o,fw} \tag{10}$$

[1]After projection to the appropriate dimension.

Let S be the set of all slots in the dataset. We define a set of $2|S|+1$ labels (one B- and I- label for each slot and a single O label) for IOB tagging. The refined token embedding $s^t_{o,m}$ is used to predict the distribution across all IOB labels for token at index m as

$$p^t_{s,m} = softmax(W_s \cdot s^t_{o,m} + b_s) \tag{11}$$

where $\dim(p^t_{s,m}) = 2|S|+1$, $W_s \in \mathbb{R}^{d_s \times 2|S|+1}$ and $b_s \in \mathbb{R}^{2|S|+1}$, $d_s = \dim(s^t_{o,m})$ is the output size of slot tagger LSTM. During inference, we predict $argmax(p^t_{s,m})$ as the slot label for the m^{th} token in the user utterance in turn t.

3.7 Updating Candidate Set

A candidate set C^t_s is defined as a set of values of a slot s which have been mentioned by either the user or the system till turn t. Rastogi et al. 2017 proposed the use of candidate sets in DST for efficiently handling slots with a large set of values. In their setup, the candidate set is updated at every turn to include new values and discard old values when it reaches its maximum capacity. The dialogue state is represented as a set of distributions over value set $V^t_s = C^t_s \cup \{\delta, \phi\}$ for each slot $s \in S^t$, where δ and ϕ are special values `dontcare` (user is ok with any value for the slot) and `null` (slot not specified yet) respectively, and S^t is the set of all slots that have been mentioned either by the user or the system till turn t.

Our model uses the same definition and update rules for candidate sets. At each turn we use the predictions of the slot tagger (Section 3.6) and system acts which having slot and value parameters to update the corresponding candidate sets. All candidate sets are padded with dummy values for batching computations for all slots together. We keep track of valid candidates by defining indicator features $m^t_v(s,c)$ for each candidate, which take the value 1.0 if candidate is valid or 0.0 if not.

3.8 Candidate Scorer

The candidate scorer predicts the dialogue state by updating the distribution over the value set V^t_s for each slot $s \in S^t$. For this, we define three intermediate features r^t_{utt}, $r^t_{slot}(s)$ and $r^t_{cand}(s,c)$. r^t_{utt} is shared across all value sets and is defined by equation 12. $r^t_{slot}(s)$ is used to update scores for V^t_s and is defined by equation 13. Furthermore, $r^t_{cand}(s,c)$ is defined for each candidate $c \in C^t_s \subset V^t_s$ using equation 14 and contains all features that are associated to candidate c of slot s.

$$r^t_{utt} = d^t_o \oplus a^t_{utt} \tag{12}$$

$$r^t_{slot}(s) = a^t_{slot}(s) \oplus [p^{t-1}_\delta(s), p^{t-1}_\phi(s)] \tag{13}$$

$$\begin{aligned} r^t_{cand}(s,c) = a^t_{cand}(s,c) \oplus [p^{t-1}_c(s)] \oplus \\ [m^t_v(s,c),\, m^t_u(c)] \end{aligned} \tag{14}$$

In the above equations, d^t_o is the dialogue context at turn t output by the state encoder (Section 3.3), a^t_{utt}, $a^t_{slot}(s)$ and $a^t_{cand}(s,c)$ are system act encodings generated by the system act encoder (Section 3.1), $p^{t-1}_\delta(s)$ and $p^{t-1}_\phi(s)$ are the scores associated with `dontcare` and `null` values for slot s respectively. $p^{t-1}_c(s)$ is the score associated with candidate c of slot s in the previous turn and is taken to be 0 if $c \notin C^t_s$. $m^t_v(s,c)$ are variables indicating whether a candidate is valid or padded (Section 3.8). We define another indicator feature $m^t_u(c)$ which takes the value 1.0 if the candidate is a substring of the user utterance in turn t or 0.0 otherwise. This informs the candidate scorer which candidates have been mentioned most recently by the user.

$$r'^t_{slot}(s) = r^t_{utt} \oplus r^t_{slot}(s) \tag{15}$$

$$l^t_s(\delta) = FF^1_{cs}(r'^t_{slot}(s)) \tag{16}$$

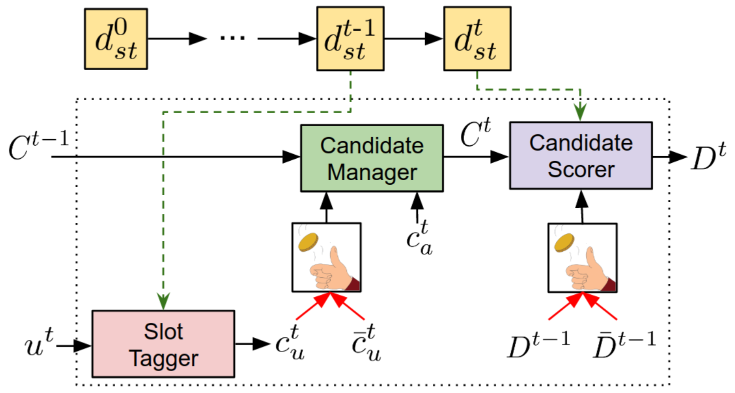
$$l^t_s(c) = FF^2_{cs}(r'^t_{slot}(s) \oplus r^t_{cand}(s,c)) \tag{17}$$

$$p^t_s = softmax(l^t_s) \tag{18}$$

Features used in Equations 12-14 are then used to obtain the distribution over V^t_s using Equations 15-17. In the above equations, $l^t_s(\delta)$ denotes the logit for `dontcare` value for slot s, $l^t_s(c)$ denotes the logit for a candidate $c \in C^t_s$ and $l^t_s(\phi)$ is a trainable parameter. These logits are obtained by processing the corresponding features using feed-forward neural networks FF^1_{cs} and FF^2_{cs}, each having one hidden layer. The output dimension of these networks is 1 and the dimension of the hidden layer is taken to be half of the input dimension. The logits are then normalized using softmax to get the distribution p^t_s over V^t_s.

4 Scheduled Sampling

DST is a recurrent model which uses predictions from the previous turn. For stability during training, ground truth predictions from the previous turn are used. This causes a mismatch between training and inference behavior. We use scheduled sampling (Bengio et al., 2015) to bridge this



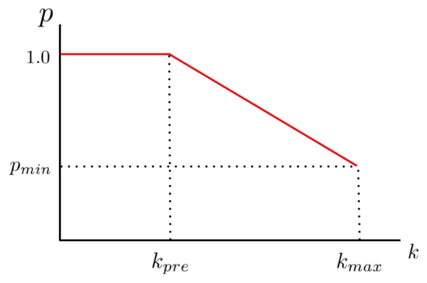


Figure 4: Illustration of scheduled sampling for training the candidate scorer. The left figure shows the two locations in our setup where we can perform scheduled sampling, while the plot on the right shows the variation of sampling probabilities p_c and p_D with training step. See Section 4 for details.

mismatch. Scheduled sampling has been shown to achieve improved slot tagging performance on single turn datasets (Liu and Lane, 2016). Figure 4 shows our setup for scheduled sampling for DST, which is carried out at two different locations - slot tags and dialogue state.

The performance of slot tagger is critical to DST because any slot value missed by the slot tagger will not be added to the candidate set (unless it is tagged in another utterance or present in any system act). To account for this, during training, we sample between the ground truth slot tags ($\bar{c}^t_u$) and the predicted slot tags (c^t_u), training initially with $\bar{c}^t_u$ (i.e. with keep probability $p_c = 1$) but gradually reducing p_c i.e. increasingly replacing $\bar{c}^t_u$ with c^t_u. Using predicted slot tags during training allows DST to train in presence of noisy candidate sets.

During inference, the candidate scorer only has access to its own predicted scores in the previous turn (Equations 13 and 14). To better mimic this setup during training, we start with using ground truth previous scores taken from $\bar{D}^{t-1}$ (i.e. with keep probability $p_D = 1$) and gradually switch to D^{t-1}, the predicted previous scores, reducing p_D.

Both p_c and p_D vary as a function of the training step k, as shown in the right part of Figure 4; only ground truth slot tags and dialogue state are used for training i.e. p_c and p_D stay at 1.0 for the first k_{pre} training steps, and then decrease linearly as the ground truth slot tags and state are increasingly replaced by model predictions during training.

5 Experiments

The major contributions of our work are two-fold. First, we hypothesize that joint modeling of LU and DST results in a computationally efficient model with fewer parameters without compromising performance. Second, we propose the use of scheduled sampling to improve the robustness of DST during inference. To this end, we conduct experiments across the following two setups.

Separate vs Joint LU-DST - Figure 3 shows the joint LU-DST setup where parameters in the utterance encoder and state encoder are shared across LU tasks (intent classification, dialogue act classification and slot tagging) and DST (candidate scoring). As baselines, we also conduct experiments where LU and DST tasks use separate parameters for utterance and state encoders.

Scheduled Sampling - We conduct scheduled sampling (as described in Section 4) experiments in four different setups.

1. *None* - Ground truth slot tags ($\bar{c}^t_u$) and previous dialogue state ($\bar{D}^{t-1}$) are used for training.
2. *Tags* - Model samples between ground truth ($\bar{c}^t_u$) and predicted (c^t_u) slot tags, sticking to ground truth previous state.
3. *State* - Model samples between ground truth ($\bar{D}^{t-1}$) and predicted (D^{t-1}) previous state, sticking to ground truth slot tags.
4. *Both* - Model samples between $\bar{D}^{t-1}$ and D^{t-1} as well as between $\bar{c}^t_u$ and c^t_u.

In the last three setups, we start sampling from predictions only after $k_{pre} = 0.3\, k_{max}$ training steps, as shown in Figure 4.

5.1 Evaluation Metrics

We report user intent classification accuracy, F1 score for user dialogue act classification, frame accuracy for slot tagging and joint goal accuracy and slot F1 score for DST. During DST evaluation, we always use the predicted slot values and the dialogue state in the previous turn. Slot frame accuracy is defined as the fraction of turns for which all slot labels are predicted correctly. Similarly, joint goal accuracy is the fraction of turns for which the predicted and ground truth dialogue state match for all slots. Since it is a stricter metric than DST slot F1, we use it as the primary metric to identify the best set of parameters on the validation set.

5.2 Datasets

We evaluate our approaches on two datasets:

- **Simulated Dialogues**[2] - The dataset, described in Shah et al. 2017, contains dialogues from restaurant (Sim-R) and movie (Sim-M) domains across three intents. A challenging aspect of this dataset is the prevalence of OOV entities e.g. only 13% of the movie names in the dev/test sets also occur in the training data.
- **DSTC2** - We use the top ASR hypothesis and system dialogue acts as inputs. Dialogue act labels are obtained from top SLU hypothesis and state labels for requestable slots. DS labels are obtained from state labels for informable slots. We use a semantic dictionary (Henderson et al., 2014) to obtain ground truth slot tags. We also use the semantic dictionary to canonicalize the candidate values since the slot values in the dialogue state come from a fixed set in the DSTC2 dialogues and may be different from those present in the user utterance.

5.3 Training

We use sigmoid cross entropy loss for dialogue act classification and softmax cross entropy loss for all other tasks. During training, we minimize the sum of all task losses using ADAM optimizer (Kingma and Ba, 2014), for 100k training steps with batches of 10 dialogues each. We used grid-search to identify the best hyperparameter values (sampled within specified range) for learning rate (0.0001 - 0.005) and token embedding dimension (50 - 200). For scheduled sampling experiments, the minimum keep rate i.e. p_{min} is varied between 0.1 - 0.9 with linear decay. The layer sizes for the utterance encoder and slot tagger are set equal to the token embedding dimension, and that of the state encoder to half this dimension.

Slot Value dropout - To make the model robust to OOV tokens arising from new entities not present in the training set, we randomly replace slot value tokens in the user utterance with a special OOV token with a probability that linearly increases from 0.0 to 0.4 during training.

6 Results and Discussion

Table 1 shows our results across the two setups described in Section 5, for the Simulated Dialogues datasets. For Sim-R + Sim-M, we observe that the joint LU-DST model with scheduled sampling (SS) on both slot tags and dialogue state performs the best, with a joint goal accuracy of 73.8% overall, while the best separate model gets a joint goal accuracy of 71.9%, using SS only for slot tags. Even for the no-SS baselines, the joint model performs comparably to the separate model (joint goal accuracies of 68.6% and 68.7% respectively), indicating that sharing results in a more efficient model with fewer parameters, without compromising overall performance. For each SS configuration, our results comparing separate and joint modeling are statistically significant, as determined by the McNemar's test with $p < 0.05$.

On the Sim-R dataset, the best joint model obtains a joint goal accuracy of 87.1%, while the best separate model obtains 85.0%. However, we observe a significant drop in joint goal accuracy for the Sim-M dataset for both the joint model and the separate model as compared to Sim-R. This can partly be attributed to the Sim-M dataset being much smaller than Sim-R (384 training dialogues as opposed to 1116) and that the high OOV rate of the *movie* slot in Sim-M makes slot tagging performance more crucial for Sim-M. While SS does gently bridge the gap between training and testing conditions, its gains are obscured in this scenario possibly since it is very hard for DST to recover from a slot value being completely missed by LU, even when aided by SS.

For the two datasets, we also observe a significant difference between the slot frame accuracy and joint goal accuracy. This is because an LU error penalizes the slot frame accuracy for a single turn, whereas an error in dialogue state propagates through all the successive turns, thereby drasti-

[2]Dataset available at http://github.com/google-research-datasets/simulated-dialogue/

Table 1: Experiments and results on test set with variants of scheduled sampling on separate and joint LU-DST models, when trained on Sim-M + Sim-R.

Eval Set	**SS Setup**	**Intent Accuracy**		**Dialogue Act F1 Score**		**Slot Frame Accuracy**		**Joint Goal Accuracy**		**DST Slot F1 Score**	
		Sep	Joint	Sep	Joint	Sep	Joint	Sep	Joint	Sep	Joint
Sim-R	None	0.999	0.997	0.956	0.935	0.924	0.919	0.850	0.846	0.951	0.952
	Tags	0.998	0.998	0.936	0.957	0.917	0.922	0.805	0.871	0.936	0.962
	State	0.999	0.998	0.931	0.939	0.919	0.920	0.829	0.852	0.935	0.951
	Both	0.994	0.998	0.948	0.919	0.917	0.916	0.829	0.849	0.942	0.953
Sim-M	None	0.991	0.993	0.966	0.966	0.801	0.800	0.276	0.283	0.806	0.817
	Tags	0.993	0.994	0.970	0.967	0.895	0.801	0.504	0.262	0.839	0.805
	State	0.996	0.970	0.964	0.955	0.848	0.799	0.384	0.266	0.803	0.797
	Both	0.989	0.996	0.970	0.959	0.887	0.860	0.438	0.460	0.805	0.845
Sim-R + Sim-M	None	0.996	0.996	0.959	0.944	0.890	0.885	0.687	0.686	0.902	0.906
	Tags	0.996	0.997	0.946	0.960	0.910	0.888	0.719	0.698	0.902	0.905
	State	0.996	0.990	0.940	0.943	0.899	0.886	0.702	0.683	0.897	0.899
	Both	0.993	0.997	0.954	0.931	0.909	0.900	0.717	**0.738**	0.894	**0.915**

cally reducing the joint goal accuracy. This gap is even more pronounced for Sim-M because of the poor performace of slot tagger on *movie* slot, which is often mentioned by the user in the beginning of the dialogue. The relatively high values of overall DST slot F1 for Sim-M for all experiments also corroborates this observation.

Table 2: Reported joint goal accuracy of model variants on the DSTC2 test set.

Model	**Separate**	**Joint**
No SS	0.661	0.650
Tags only SS	0.655	**0.670**
State only SS	0.661	0.660
Tags + State SS	0.656	0.658
Liu and Lane 2017	-	0.73
Mrkšić et al. 2017	-	0.734

Table 2 shows our results on the DSTC2 dataset, which contains dialogues in the restaurant domain. The joint model gets a joint goal accuracy of 65.0% on the test set, which goes up to 67.0% with SS on slot tags. Approaches like NBT (Mrkšić et al., 2017) or Hierarchical RNN (Liu and Lane, 2017) are better suited for such datasets, where the set of all slot values is already known, thus eliminating the need for slot tagging. On the other hand, our setup uses slot tagging for candidate generation, which allows it to scale to OOV entities and scalably handle slots with a large or unbounded set of possible values, at the cost of performance.

Analyzing results for scheduled sampling, we observe that for almost all combinations of metrics, datasets and joint/separate model configurations, the best result is obtained using a model trained with some SS variant. For instance, for Sim-M, SS over slot tags and state increases joint goal accuracy significantly from 28.3% to 46.0% for joint model. SS on slot tags helps the most with Sim-R and DSTC2: our two datasets with the most data, and low OOV rates, while SS on both slot tags and dialogue state helps more on the smaller Sim-M. In addition, we also found that slot value dropout (Section 5.3), improves LU as well as DST results consistently. We omit the results without this technique for brevity.

7 Conclusions

In this work, we present a joint model for language understanding (LU) and dialogue state tracking (DST), which is computationally efficient by way of sharing feature extraction layers between LU and DST, while achieving an accuracy comparable to modeling them separately across multiple tasks. We also demonstrate the effectiveness of scheduled sampling on LU outputs and previous dialogue state as an effective way to simulate inference-time conditions during training for DST, and make the model more robust to errors.

References

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Sequential dialogue context modeling for spoken language understanding. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 103–114.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Yun-Nung Chen, Dilek Hakkani-Tür, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dilek Hakkani-Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm.

M. Henderson, B. Thomson, and S. Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Bing Liu and Ian Lane. 2016. Joint online spoken language understanding and language modeling with recurrent neural networks. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 22.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Proceedings of Interspeech*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1777–1788.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 305–314.

A. Rastogi, D. Hakkani-Tür, and L. Heck. 2017. Scalable multi-domain dialogue state tracking. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

P. Shah, D. Hakkani-Tür, G. Tur, A. Rastogi, A. Bapna, N. Nayak, and L. Heck. 2017. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. 2018. How time matters: Learning time-decay attention for contextual spoken language understanding in dialogues. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

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 *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432.

Jason Williams. 2013. Multi-domain learning and generalization in dialog state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 433–441.

Jason D Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291.

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