Dialogue & Discourse 7(3) (2016) 47–64 doi: 10.5087/dad.2016.302

Dialog History Construction with Long-Short Term Memory for Robust Generative Dialog State Tracking

Byung-Jun Lee BJLEE@AI.KAIST.AC.KR
School of Computing, College of Engineering
Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

Kee-Eung Kim KEKIM@CS.KAIST.AC.KR
School of Computing, College of Engineering
Korea Advanced Institute of Science and Technology
291 Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea

Editor: Jason D. Williams, Antoine Raux, and Matthew Henderson

Submitted 04/15; Accepted 02/16; Published online 04/16

Abstract

One of the crucial components of dialog system is the dialog state tracker, which infers user's intention from preliminary speech processing. Since the overall performance of the dialog system is heavily affected by that of the dialog tracker, it has been one of the core areas of research on dialog systems. In this paper, we present a dialog state tracker that combines a generative probabilistic model of dialog state tracking with the recurrent neural network for encoding important aspects of the dialog history. We describe a two-step gradient descent algorithm that optimizes the tracker with a complex loss function. We demonstrate that this approach yields a dialog state tracker that performs competitively with top-performing trackers participated in the first and second Dialog State Tracking Challenges.

Keywords: Dialog State Tracking, Dialog Management, Spoken Dialog System (SDS), Hidden Information State (HIS), Long Short Term Memory (LSTM).

1. Introduction

Spoken dialog systems (SDS) are a rapidly growing subject of research with the grand challenge of developing intelligent systems capable of understanding conversational commands by human users. The generic architecture of a spoken dialog system is shown in Figure 1: the system decodes the user voice into the list of hypotheses in a form of action type and slot-value pairs (see Table 1 for details), which is defined as the spoken language understanding (SLU). The system then analyzes the hypotheses to decide how to respond to the user, which is referred to as dialog management (DM). The DM can be further decomposed into two sub-tasks: the dialog state tracking that identifies the user goal and the dialog policy optimization that finds the appropriate response. The actual vocal response is produced by the speech generation.

Due to the inevitable misinterpretation of the user utterance (*i.e.* ambiguity) in SLU, robust DM is essential in any dialog system. Since the task of determining an appropriate response based on the history of SLU data aligns well with reinforcement learning framework, a large number of previous studies were based on the Markov decision process (MDP, Levin et al. 1998; Daubigney

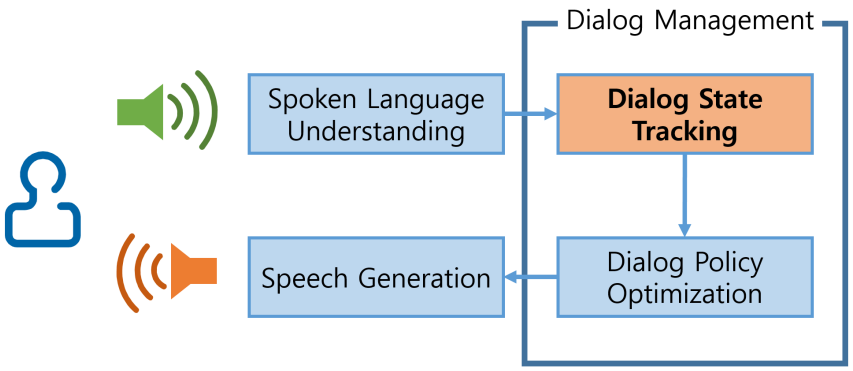

Figure 1: A diagram of general spoken dialog systems

et al. 2012) and the partially observable Markov decision process (POMDP, Williams and Young 2007; Young et al. 2010; Gasic and Young 2014). One of the most representative work is the SDS-POMDP (Young, 2006). It treats SLU data as noisy and partial observations of the underlying state of dialog that encapsulates sufficient information for DM: user's intention, utterance and the dialog history. Actions of the SDS-POMDP are defined as the abstract system responses, one of which will be selected to maximize the expected accumulation of rewards. In this fashion, the SDS-POMDP provides a unified decision theoretic model for DM.

Given a pre-determined dialog strategy, SDS-POMDP tracks the dialog state using a Bayesian filter. This is often referred to as a generative dialog state tracker, which facilitates identifying core components that can be engineered individually through standard statistical machine learning techniques. However, the performance of the optimized tracker still critically depends on what information is captured from dialog for bookkeeping. If an important aspect of the dialog is not captured and kept appropriately, it is evident that the tracker would not be so accurate in inferring the user's intention. In most of the previous work, this bookkeeping was done through heuristically selecting the information and embedding it into the dialog history (Young et al., 2010).

On the other hand, we note that there are a number of recent approaches using discriminative models, which do not involve explicit bookkeeping mechanism (Metallinou et al., 2013; Lee, 2013; Williams, 2014). They use features that are defined over the finite window of previous dialog turns. A more recent work using recurrent neural network (Henderson et al., 2014b) enables learning with features that are defined over the window of arbitrary length. In these approaches, the bookkeeping mechanism can be viewed as implicit and parameterized, which is optimized through training. This in general made the tracker perform much better than generative ones.

In this paper, we present a generative dialog state tracker that uses Long-Short Term Memory (LSTM), which is one of the deep-learning architectures for recurrent neural network that overcomes the major drawback of original recurrent neural network called *vanishing gradients* (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997). With this approach, we can preserve the intuitiveness and compositionality of generative dialog state trackers while autonomously bookkeeping appropriate information over time, which results a boost in performance. This is especially useful in practical applications, where we typically have some prior knowledge on the characteristics of the dialog domain and speech and language processing unit.

Our method extends the Bayesian filter in SDS-POMDP so that the dialog history is replaced with the embedding vector computed by LSTM. We then propose a two-step optimization algorithm to deal with the local convergence when training the model. Our tracker performed better than any tracker from the Dialog State Tracking Challenge 1 (DSTC1, Williams et al. 2013)[1] and on par with top-performing trackers from the Dialog State Tracking Challenge 2 (DSTC2, Henderson et al. 2014a)[2]. This paper is an extension of Kim et al. (2013); Lee et al. (2014) using LSTM for the dialog history.

The rest of the paper is organized as follows. Section 2 describes the Bayesian filtering model of the SDS-POMDP, with the additional techniques we adopted in this work. Section 3 describes the component probability models outlined in section 2. The optimization algorithm designed for our tracker is explained in section 4. In section 5, the experiment setup, the result and analyses are described. We conclude the paper with discussions in section 6.

2. Bayesian Filtering for the Dialog State Tracking

We start from the basic setting of the SDS-POMDP framework: in each turn of the dialog, the system executes system action a, and the user with goal g responds to the system with user action u. The SLU module processes the user action and generates the result as an N-best list (observation, in other words) $\boldsymbol{o} = [\langle \tilde{u}_1, f_1 \rangle, \ldots, \langle \tilde{u}_N, f_N \rangle]$ of the hypothesized user action $\tilde{u}_i$ and its associated confidence score f_i. Table 1 shows a dialog where the SLU generates N-best lists from noisy user actions and the system responds to the user after tracking true user goals from the SLU output.

Here, the dialog state in each turn is defined as $s = (u, g, h)$, where h is the dialog history encapsulating additional information needed for tracking the dialog state (Williams, 2008). Because the system cannot exactly identify the user goal, it maintains a posterior probability distribution over user goals, called a *belief*. The belief over the dialog states is updated by Bayesian filtering:

$$b'(u', g', h') = \eta \Pr(\boldsymbol{o}|u', g', h', a', a) \sum_{u,g,h} \Pr(u', g', h'|u, g, h, a, a') b(u, g, h) \tag{1}$$

where η is the normalization constant. With reasonable independence assumptions, the joint probabilities in the equation can be factorized and approximated into simpler forms. Further assuming that the user goal does not change, we obtain

$$\Pr(\boldsymbol{o}|u', g', h', a', a) \approx \Pr(\boldsymbol{o}|u') \tag{2}$$

$$\begin{aligned} \Pr(u', g', h'|u, g, h, a', a) &\approx \Pr(u'|g', h', a') \Pr(g'|u', g, h, a) \Pr(h'|u, g, h, a) \\ &\approx \Pr(u'|g', h', a') \Pr(h'|u, g, h, a) \mathbb{I}(g = g') \end{aligned} \tag{3}$$

$$b'(u', g', h') = \eta \Pr(\boldsymbol{o}|u') \Pr(u'|g', h', a') \sum_{u,h} \Pr(h'|u, g, h, a) b(u, g, h) \tag{4}$$

where $\Pr(\boldsymbol{o}|u')$ is the *observation model*, $\Pr(u'|g', h', a)$ is the *user action model*, $\Pr(h'|u, g, h, a)$ is the *history model*. According to the formulation, the posterior computation has to be carried out for *all* possible user goals in order to obtain the normalizing constant η. This is not feasible for

1. http://research.microsoft.com/en-us/events/dstc
2. http://camdial.org/ mh521/dstc/

Table 1: An example of a dialog depicted in DSTC1

System	East Pittsburg Bus Schedules. Say a bus route, like 28X, or say I'm not sure. hello(), request(route), example(route=28x), example(route=dont_know)
User SLU	61A 0.77 inform(route=61a) 0.12 inform(route=61) 0.01 inform(route=61d)
System	Okay, 61A. To change, say go back. Where are you leaving from? impl-conf(route=61a), example(act=goback), request(from)
User SLU	Downtown 0.59 inform(from.desc=downtown) 0.10 inform(from.desc=from downtown)
System	Okay, downtown. You can always say go back. And where are you going to? impl-conf(from.desc=downtown), example(act=goback), request(to)
User SLU	East Pittsburgh East Pittsburgh 0.25 inform(to.desc=pittsburgh) 0.20 inform(to.desc=east pittsburgh) 0.01 inform(to.desc=to pittsburgh)
System	Pittsburgh. Is that right? expl-conf(to.desc=pittsburgh)
User SLU	No 0.71 negate() 0.07 affirm()
System	Oh, sorry, where are you going to? sorry(), request(to)
User SLU	East Pittsburgh 0.97 inform(to.desc=east pittsburgh)
System	Okay, East Pittsburgh. Do you want times for the next few buses? Say yes or no. impl-confirm(to.desc=east pittsburgh) , expl-confirm(act=want-next-bus) example(act=affirm), example(act=negate)
User SLU	Yes 0.97 affirm()
System	Ok, at 7:45 PM today, there is a 61 A, from Fifth Avenue And Market Street, Downtown. schedule(result.from.time.hour=7, result.from.time.minute=45, result.from.time.ampm=pm, result.from.date.day=today, result.route=61a, result.from.desc=fifth avenue and market street, downtown

real dialog domains that have a large number of user goals as in DSTC1 and DSTC2 domains. As such, we adopt the goal partitioning approach in HIS (Young et al., 2010) that groups user goals into equivalence classes and assigns a single probability for each class. The equivalence classes are called partitions and denoted by ψ.

Beginning with one root partition with the probability 1, partitions are split whenever the distinction is required by observations, *i.e.* a user action hypothesis from the SLU output. This confines

the possible goal states to the values that have been appeared at least once as an SLU hypothesis, and provides scalability without a significant loss in accuracy when the coverage of N-best list is extensive enough to include the true user action. By defining belief refinement model $\Pr(\psi'|\psi)$ to be a probability mass ratio that the partition splits, the Bayesian filtering equation (4) becomes:

$$b'(u', \psi', h') = \eta \Pr(\boldsymbol{o}|u') \Pr(u'|\psi', h', a') \sum_{u,h} \Pr(h'|u, \psi, h, a) \Pr(\psi'|\psi) b(u, \psi \supset \psi', h) \quad (5)$$

where $\psi \supset \psi'$ denotes that ψ is a parent partition (superset) of ψ'.

Even with the partitioned approach, the total number of partitions grows very fast as dialog turn progresses (exponential in number of observed goal types, polynomial in number of observed goals), and it is required to give certain limit to the number of partitions. We used the incremental partition recombination algorithm (Williams, 2010), which recombines the less important partitions in each turn if the number of partitions exceeds the threshold.

3. Designing Component Probability Models

The equation (5) needs a model for each component probability term, i.e. observation model, user action model, history model, and belief update model. Although assigning simple statistics to each component probability model (Young et al., 2010; Kim et al., 2013) can be a valid choice, Lee et al. (2014) has shown that designing each probabilities appropriately and optimizing it results in much better performance. In following subsections, each component probability model is described.

3.1 Observation model

The observation model $\Pr(\boldsymbol{o}|u)$ represents the probability of structured observation $\boldsymbol{o} = [\langle \tilde{u}_1, f_1 \rangle, \ldots, \langle \tilde{u}_N, f_N \rangle]$ given user action u. Here, we assume that the observation model only depends on the type of user action u and its associated confidence score f_i to obtain a simple model. Then, we obtain

$$\Pr(\boldsymbol{o}|u = \tilde{u}_i) = k \cdot (\exp(\boldsymbol{w}_{type(\tilde{u}_i)}) f_i + \exp(b_{type(\tilde{u}_i)})) \quad (6)$$

where $\boldsymbol{w}_{type(u)}$ is the weight associated with specific user action type of u and $b_{type(u)}$ is bias term for the same user action type, which are exponentiated for preserving non-negativity. k is the normalizing constant, which can be ignored since it is subsumed by the constant η in the belief update equation (5).

The user action type here is domain-specific attribute of SLU hypothesis, which is given in prior (*e.g. inform*, *affirm*, *negate* in Table 1). Refer to DSTC1 or DSTC2 handbook for full description of user and system action types.

3.2 Belief refinement model

The belief refinement model $\Pr(\psi'|\psi)$ defines the split ratio of partitions when the partition is required to split due to the observations. According to the definition, the full specification of the belief refinement model corresponds to the prior distribution of partitions, which is summation of prior probability of each individual user goals included in partitions. We define the prior on goals $\Pr(g)$ as a smoothed empirical distribution

$$\Pr(g) = \epsilon \frac{\#turns\ having\ goal\ g}{\#turns} + (1 - \epsilon) \frac{1}{\#individual\ goals} \quad (7)$$

With the prior probability of each individual user goals, we can compute belief refinement model by summing the prior probabilities of user goals to get the ratio

$$\Pr(\psi'|\psi) = \frac{\sum_{g' \in \psi'} \Pr(g')}{\sum_{g \in \psi} \Pr(g)} \tag{8}$$

if the ψ is divided to ψ' and other partitions this turn by observation, $\Pr(\psi'|\psi) = 0$ otherwise.

3.3 User action model and history model

In previous approaches (Young et al., 2010; Lee et al., 2014), the user action model $\Pr(u'|\psi', h', a')$ only utilized the features of current turn and relied on manually defined bookkeeping methods for h'. This is potentially disadvantageous compared to trackers that are able to learn which information to maintain over multiple dialog turns (Metallinou et al., 2013; Lee, 2013; Williams, 2014; Henderson et al., 2014b).

This disadvantage becomes apparent in complex domains like DSTC2, where the conversation topics change over time. In DSTC2, in which the domain was a restaurant information system, the user usually starts with providing search constraints and then proceeds to the stage of choosing the restaurant to be informed. Learning such phase transition in dialogs where the user acts differently is therefore crucial for designing robust user action model.

Recurrent neural networks provide a natural model for handling such problems, as they are able to model any dynamic sequences with complex behaviors by learning to choose what information to keep. Adopting recurrent neural network in dialog state tracking has proven its success in Henderson et al. (2014b), while our approach is more focused on learning the dependencies over dialog turns in a generative state tracker, encoding them as the dialog history. Instead of the conventional recurrent neural network, we adopt one of the variants, namely Long-Short Term Memory (LSTM, Bengio et al. 1994; Hochreiter and Schmidhuber 1997), which effectively mitigates the *vanishing gradient* problem in training recurrent neural networks.

3.3.1 LSTM ARCHITECTURE USED FOR USER ACTION MODEL

The input to LSTM is a feature constructed from u, ψ and a. We used every combinations of [*user action type, system action type, partition-action consistency*] that appears in the training set as a feature set, which results in maximum 1 of $|u| \cdot |a| \cdot 4$ coding[3].

In the following equations of LSTM,

- $\phi(u_t, \psi_t, a_t)$ is 1 of $|u| \cdot |a| \cdot 4$ coding feature vector, which is the input to LSTM.
- C_{t-1}, C_t are the cell values stored in LSTM.
- y_{t-1}, y_t are the outputs of LSTM, and also stored.
- $W_p, W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o, V_i, V_f, V_o, w_d, w_l$ are weight matrices (or vectors) to be learned.
- b_i, b_f, b_c, b_o are bias vectors.

3. Since we only considered the feature combinations that appear in the training set, the vector length is much less than $|u| \cdot |a| \cdot 4$ because of the nonexistent *system action-user action* combination. Also, due to the dual or more action types, there are the cases where it is not exactly 1-of-k coding.

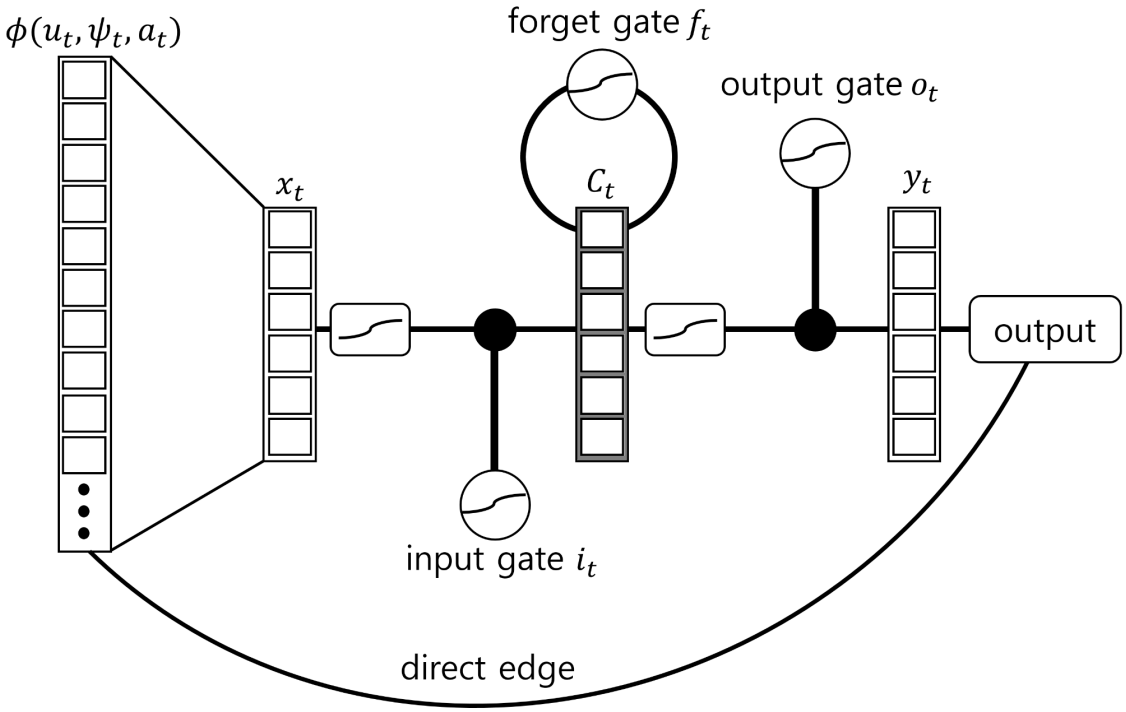

Figure 2: The Long-Short Term Memory architecture used to model user action model. From the usual LSTM architecture, a linearly compressing input layer and direct edge from input to output is added. The gray colored cell vector is maintained through the time.

The input layer is passed through the input gate to compute a new cell value. The cell value is computed by adding input and the previous cell if not forgotten, as follows.

$$i_t = \sigma(W_i\phi(u, \psi_t, a_t) + U_i y_{t-1} + V_i C_{t-1} + b_i) \tag{9}$$
$$\tilde{C}_t = tanh(W_c\phi(u, \psi_t, a_t) + U_c y_{t-1} + b_c) \tag{10}$$
$$f_t = \sigma(W_f\phi(u, \psi_t, a_t) + U_f y_{t-1} + V_f C_{t-1} + b_f) \tag{11}$$
$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \tag{12}$$

where $*$ denotes the elementary multiplication between vectors. Output is then computed by the output gate:

$$o_t = \sigma(W_o\phi(u, \psi_t, a_t) + U_o y_{t-1} + V_o C_t + b_o) \tag{13}$$
$$y_t = o_t * tanh(C_t) \tag{14}$$

We finally get the user action model value $\Pr(u_t|\psi_t, h_t, a_t)$ by adding the direct edge from feature function and using a softmax function,

$$\Pr(u_t|\psi_t, h_t, a_t) = \frac{\exp(w_l^T y_t(\phi(u_t, \psi_t, a_t)) + w_d^T \phi(u_t, \psi_t, a_t))}{\sum_u \exp(w_l^T y_t(\phi(u, \psi_t, a_t)) + w_d^T \phi(u, \psi_t, a_t))}. \tag{15}$$

where y_t on the right hand side of the equation is implicitly dependent on h_t via the information stored in LSTM. In practice, however, we cannot sum up the LSTM outputs for every u. Instead, we only evaluate the LSTM for $\tilde{u}_i$s from N-best list, and evaluate $\tilde{u}_{offlist}$ for the remaining actions.

3.3.2 HISTORY MODEL AND M-BEST LSTMS

According to the model above, what we are maintaining as a history state h_t is the cell layer and the output layer of the last turn's LSTM, in which the sequence $[u_1, a_1, \psi_1, \ldots, u_{t-1}, a_{t-1}, \psi_{t-1}]$ is implicitly embedded. The history model $\Pr(h_t|u_{t-1}, \psi_{t-1}, h_{t-1}, a_{t-1})$ is therefore deterministic, and the Bayesian filtering equation is then

$$\begin{aligned} b'(\psi', h') &= \eta \sum_{u'} \Pr(\boldsymbol{o}|u') \Pr(u'|\psi', h', a') \sum_{h,u} \Pr(h'|u, \psi, h, a) \Pr(\psi'|\psi) b(u, \psi \supset \psi', h) \\ &= \eta \sum_{u'} \Pr(\boldsymbol{o}|u') \Pr(u'|\psi', h', a') \sum_{h,u} \mathbb{I}(h' = [u, \psi, a, h]) \Pr(\psi'|\psi) b(u, \psi \supset \psi', h) \\ &= \eta \sum_{u'} \Pr(\boldsymbol{o}|u') \Pr(u'|\psi', h', a') \Pr(\psi'|\psi) b(\psi \supset \psi', h \supset h') \end{aligned} \tag{16}$$

where $h \supset h'$ denotes that h is a parent history of h' (*i.e.* $h' = \{h, a, u, \psi\}$). This implies that we should maintain $|h||\psi|$ partitions and $|h|$ LSTMs each turn to exactly compute posterior probability over user goals. However, it is impractical due to the exponentially increasing size of history. To gain a better insight, the following equation is what to be evaluated for the dialog state tracking problem:

$$b'(\psi') = \eta \sum_{u'} \Pr(\boldsymbol{o}|u') \Pr(\psi'|\psi) \sum_{h'} \Pr(u'|\psi', h', a') b(\psi \supset \psi', h \supset h') \tag{17}$$

It can be seen that we are taking the weighted average of LSTMs over the joint belief $b(u \in h', \psi \supset \psi', h \supset h')$ for every h'. Similar to the partition recombination technique (Williams, 2010) that limits the number of partition to certain constant, we can approximate it by limiting the number of histories by M (equivalently, M LSTMs). Since the aggregation of histories are not possible, other than M histories with the best belief are ignored.

A belief update with M-LSTMs is performed, as specified in the following algorithm 1.

Data: system action a, observation $\boldsymbol{o} = \{(\tilde{u}_1, f_1), ..., (\tilde{u}_N, f_N)\}$,
partitions $\Psi = \{\psi_1, ...\psi_P\}$, LSTMs $L = \{l_1, ..., l_M\}$, histories $H = \{h_1, ..., h_M\}$
and the belief $b(\psi, h)$ $\forall \psi, h$
Result: Partitions Ψ', LSTMs L', histories H' and the belief b'
$\Psi' = \{\}$;
$H' = \{\}$ **for** $\tilde{u}_i$ *in* $\boldsymbol{o}$ **do**
  for ψ_j *in* Ψ **do**
    if $\tilde{u}_i$ *need to split* ψ_j **then**
      $\Psi' = \Psi' \cup \{\psi_j \wedge \tilde{u}_i, \psi_j \wedge \neg\tilde{u}_i\}$;
      calculate $\Pr(\psi_j \wedge \tilde{u}_i | \psi_j)$ by equation (8);
      for h_k *in* H **do**
        $b(\psi_j \wedge \tilde{u}_i, h_k) = \Pr(\psi_j \wedge \tilde{u}_i | \psi_j) b(\psi_j, h_k)$;
        $b(\psi_j \wedge \neg\tilde{u}_i, h_k) = b(\psi_j, h_k) - b(\psi_j \wedge \tilde{u}_i, h_k)\ b(\psi_j, h_k) = 0$;
      end
    end
  end
end
$b'(\psi, h) = 0 \quad \forall \psi, h$;
for ψ_i *in* Ψ' **do**
  for $\tilde{u}_j$ *in* $\boldsymbol{o}$ **do**
    calculate $\Pr(\boldsymbol{o} | u = \tilde{u}_i)$ by equation (6);
    for h_k *in* H **do**
      $h'_k = [\tilde{u}_j, a, \psi_i, h_k]$;
      $H' = H' \cup \{h'_k\}$;
      calculate $\Pr(\tilde{u}_j | \psi_i, h_k, a)$ by equation (15) and LSTM l_k;
      $b'(\psi_i, h'_k) = b'(\psi_i, h'_k) + \Pr(\boldsymbol{o} | u = \tilde{u}_i) \Pr(\tilde{u}_j | \psi_i, h_k, a) b(\psi_i, h_k)$
    end
  end
end
if $|H'| > M$ **then**
  order H' by decreasing $\sum_\psi b'(\psi, h)$;
  for h_k *in* H', $k > M$ **do**
    $H' = H' - \{h_k\}$;
    $b'(\psi, h_k) = 0\ \ \forall \psi$
  end
end
$L' =$ updated L to contain embedded histories of H';
if $|\Psi'| > P$ **then**
  run partition recombination according to Williams (2010)
end
$b'(\psi, h) = \frac{b'(\psi, h)}{\sum_{\psi, h} b'(\psi, h)} \quad \forall \psi, h$

Algorithm 1: Updating the belief over partitions and histories

4. Optimization

In this paper, we use L2 metric as the loss function since it is found to be most influential to dialog system performance (Lee, 2014). The model is hence optimized to minimize the L2 loss function

$$L(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{\psi\in\Psi_{i,t}}(r(\psi)-b(\psi))^2 \tag{18}$$

where i sums over N training instances, t sums over T turns of each training instance and $\boldsymbol{\psi}_{i,t}$ is the group of partitions in the instance i, turn t. $r(\psi)$ is the binary label with value 1 if and only if the partition ψ contains the true user goal. In the rest of the section, gradient methods used to optimize with respect to the loss function are described.

4.1 Cascading gradient

Taking the insight from the BPTT algorithm (Mozer, 1989), we can unfold the update formula through time and calculate the gradient with respect to weight vectors. For instance, considering a parameter w from the observation model, $\partial L/\partial w$ can be obtained by:

$$\frac{\partial L}{\partial w} = \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{\psi\in\Psi_{i,t}}(b(\psi)-r(\psi))\frac{\partial b(\psi)}{\partial w} \tag{19}$$

$$\begin{aligned}
\frac{\partial b'(\psi')}{\partial w} &= \frac{\partial \eta}{\partial w}\sum_{u'\in\boldsymbol{o}}\Pr(\boldsymbol{o}|u')\Pr(\psi'|\psi)\sum_{h'}\Pr(u'|\psi',h',a')b(\psi\supset\psi',h\supset h') \\
&\quad+\eta\sum_{u'\in\boldsymbol{o}}\frac{\partial\Pr(\boldsymbol{o}|u')}{\partial w}\Pr(\psi'|\psi)\sum_{h'}\Pr(u'|\psi',h',a')b(\psi\supset\psi',h\supset h') \\
&\quad+\eta\sum_{u'\in\boldsymbol{o}}\Pr(\boldsymbol{o}|u')\Pr(\psi'|\psi)\sum_{h'}\Pr(u'|\psi',h',a')\frac{\partial b(\psi\supset\psi',h\supset h')}{\partial w} \\
&= \eta\sum_{u'\in\boldsymbol{o}}\frac{\partial\Pr(\boldsymbol{o}|u')}{\partial w}\Pr(\psi'|\psi)\sum_{h'}\Pr(u'|\psi',h',a')b(\psi\supset\psi',h\supset h') \\
&\quad-b(\psi')\sum_{\bar{\psi}\in\Psi_{i,t}}\eta\sum_{u'\in\boldsymbol{o}}\frac{\partial\Pr(\boldsymbol{o}|u')}{\partial w}\Pr(\psi'|\bar{\psi})\sum_{h'}\Pr(u'|\psi',h',a')b(\bar{\psi}\supset\psi',h\supset h') \\
&\quad+\eta\sum_{u'\in\boldsymbol{o}}\Pr(\boldsymbol{o}|u')\Pr(\psi'|\psi)\sum_{h'}\Pr(u'|\psi',h',a')\frac{\partial b(\psi\supset\psi',h\supset h')}{\partial w} \\
&\quad-b(\psi')\sum_{\bar{\psi}\in\Psi_{i,t}}\eta\sum_{u'\in\boldsymbol{o}}\Pr(\boldsymbol{o}|u')\Pr(\psi'|\bar{\psi})\sum_{h'}\Pr(u'|\psi',h',a')\frac{\partial b(\bar{\psi}\supset\psi',h\supset h')}{\partial w}
\end{aligned} \tag{20}$$

We call the above *cascading gradient* since $\partial b'(\psi)/\partial w$ requires computation of the gradient in the previous dialog turn $\partial b(\psi)/\partial w$, and hence reflects the temporal impact of the parameter change throughout the dialog turns. Once we obtain the gradients, we can simultaneously update all the parameters with any algorithm. In this paper, L-BFGS was used.

4.2 Initialization using simple gradient

The L-BFGS using cascading gradient, however, seriously suffers from the local convergence since the cost function is very complex in parameter space, mostly due to the repeated appearance of features throughout dialog turns. The randomized initialization had a limited effect because of the large dimensionality of parameter space.

If we ignore the gradient from previous turn and treat each turn as individual training instance, the gradient becomes, for example of a parameter w from the observation model,

$$\begin{aligned}
\frac{\partial b'(\psi')}{\partial w} &= \frac{\partial \eta}{\partial w} \sum_{u' \in \boldsymbol{o}} \Pr(\boldsymbol{o}|u') \Pr(\psi'|\psi) \sum_{h'} \Pr(u'|\psi', h', a') b(\psi \supset \psi', h \supset h') \\
&\quad + \eta \sum_{u' \in \boldsymbol{o}} \frac{\partial \Pr(\boldsymbol{o}|u')}{\partial w} \Pr(\psi'|\psi) \sum_{h'} \Pr(u'|\psi', h', a') b(\psi \supset \psi', h \supset h') \\
&= \eta \sum_{u' \in \boldsymbol{o}} \frac{\partial \Pr(\boldsymbol{o}|u')}{\partial w} \Pr(\psi'|\psi) \sum_{h'} \Pr(u'|\psi', h', a') b(\psi \supset \psi', h \supset h') \\
&\quad - b(\psi') \sum_{\bar{\psi} \in \Psi_{i,t}} \eta \sum_{u' \in \boldsymbol{o}} \frac{\partial \Pr(\boldsymbol{o}|u')}{\partial w} \Pr(\psi'|\bar{\psi}) \sum_{h'} \Pr(u'|\psi', h', a') b(\bar{\psi} \supset \psi', h \supset h') \quad (21)
\end{aligned}$$

Since it is an approximation that only catches the direct impact of the features and makes loss function much simpler, it takes the parameters close to the global optimum, increasing the chance of L-BFGS converging to the good local optimum. Our optimization algorithm is hence composed of two steps, first initializing with the simple gradient and then training with the cascading gradient.

5. Experiment and Analysis

Table 2: Description of the DSTC1 datasets used for the tracker (Bus Information System)

Dataset	Calls	Similarity	Slot
train1a	1013	Similar to train2	9
train2	678	Similar to train1a	9
train3	779	Distinct from train1a and 2	5
test1	765	Very similar to train1a and 2	9
test2	983	Similar to train1a and 2	5
test3	1037	Very similar to train3	5

In the experiments, we used datasets from both DSTC1 and DSTC2. Three labeled training datasets *(train1a, train2, train3)* and four test datasets *(test1, test2, test3, test4)* were included in DSTC1 whereas two labeled training datasets *(DSTC2_train, DSTC2_dev)* and one test dataset *(DSTC2_test)* were included in DSTC2. Description of the datasets is provided in Table 2 and Table 3. *Test4* is omitted in this paper since the significant number of missing or incorrect labels were found.

We can group the datasets based on their characteristics. Datasets *train1a, train2, test1*, and *test2* have many turns in each call. However, they include only one hypothesis in each dialog turn

Table 3: Description of the DSTC2 datasets used for the tracker (Restaurant Information System)

Dataset	Calls	Similarity	Slot
DSTC2_train	1612	Very similar to dev	4
DSTC2_dev	506	Very similar to train	4
DSTC2_test	1117	Similar to train and test	4

(1-best SLU output) and the user goal rarely changes. These datasets can be said to be relatively easy since there is not much information to examine in inferring the user goal.

On the other hand, datasets *train3, test3, DSTC2_test, DSTC2_dev* and *DSTC2_test* are considered harder. These datasets contain dialogs that are closer to real world conversation with latent relations among system actions and user actions and dialog flows. Dependencies that span over multiple dialog turns are more frequent and they even change over turns in DSTC2 datasets.

For each target test dataset, we chose the training datasets that are known to be similar. Specifically, *Train1a* and *train2* are used to train the tracker that tests *test1* and *test2*, whereas *train3* is used for *test3* and *DSTC2_train, DSTC2_dev* are used for *DSTC2_test*. We only used the SLU data for observations (i.e. ignored ASR information), and our result is therefore compared with teams that only used SLU data. Note that the evaluations of our algorithm are taken after the release of the test set while the other teams' results are evaluated before the release of the test set.

We measured the tracker performance according to the following evaluation metrics used in DSTC1[4]:

- **accuracy (acc)** measures the rate of the most likely hypothesis h_1 being correct.
- **average score (avgp)** measures the average of scores assigned to the correct hypotheses.
- **L2** follows the definition in the equation (18).
- **mean reciprocal rank (mrr)** measures the average of $1/R$, where R is the minimum rank of the correct hypothesis.
- **ROC equal error rate (eer)** is the sum of false accept (FA) and false reject (FR) rates when FA rate=FR rate.
- **ROC.v1.**P measures correct accept (CA) rate when there are at most $P\%$ false accept (FA) rate.

$N(\text{FA})$, $N(\text{CR})$, $N(\text{CA})$ and $N(\text{FR})$ are the number of false accepts (FA), correct rejects (CR), correct accepts (CA), and false rejects (FR) respectively. Additionally, N_D is the total number of data instances. The evaluation takes the most likely hypothesis h_1 and its score s_1 and compares them with the threshold θ and the ground truth user goal h^*. Each evaluation increments appropriate counters by

$$\begin{cases} N(\text{CA})\text{++} & \text{if } s_1 \geq \theta \text{ and } h_1 = h^* \\ N(\text{CR})\text{++} & \text{if } s_1 < \theta \text{ and } h_1 \neq h^* \\ N(\text{FA})\text{++} & \text{if } s_1 \geq \theta \text{ and } h_1 \neq h^* \\ N(\text{FR})\text{++} & \text{if } s_1 < \theta \text{ and } h_1 = h^* \end{cases}$$

4. http://research.microsoft.com/apps/pubs/?id=169024

The CA rate is defined as $\frac{N(\text{CA})}{N_D}$ and the FA rate is defined as $\frac{N(\text{FA})}{N_D}$. **ROC.v2.**P from the DSTC1 metrics is not included in the experiment.

With datasets, the baseline trackers that work in a simple deterministic manner was included. For example, the most basic one picks the SLU hypothesis with the highest probability so far for each goal slot. More detailed description is given in DSTC2 handbook.

5.1 DSTC1 Result

Table 4 compares the performance of our tracker against that of the most competitive entries from DSTC1 participants. Base stands for the baseline tracker and X-Y stands for team X entry Y. The following is the summary description provided by the teams:

- Team 1 : deep neural network (Henderson et al., 2013)
- Team 6 : feature-rich discriminative model (Lee and Eskenazi, 2013)
- Team 9 : generative dialog state tracker with basic statistics (Kim et al., 2013)

The following abbreviations are used for our tracker with different history models and optimization algorithms:

- C: direct edge only user action model optimized with cascading gradient.
- SC: direct edge only user action model initialized with simple gradient and optimized with cascading gradient.
- LC: LSTM history based user action model optimized with cascading gradient.
- LSC: LSTM history based user action model initialized with simple gradient and optimized with cascading gradient.

We choose the trackers with the minimum training objective function among 100 random seeds. The number of partitions is limited to 10, and the number of histories (LSTM, equivalently) is limited to 3. 50 units are used for all (input, cell, output) layers of LSTMs. This setting of parameters enable the real-time tracking of dialog state, while preserving most of the information required.

Overall, the trackers initialized with simple gradients outperform the trackers without the initialization process. It can be seen that the local convergence problem of cascading gradient is most severe in *test3*, as the problem gets more complex even though the other domains also show the clear difference in scores. The trackers optimized with two different gradients outperform the scores of other teams participated in DSTC1.

In case of history construction using LSTM, DSTC1 *test1* and *test2*, which are categorized as easy dataset, are not showing any performance gain from the history construction. This is expected due to the fact that the dialogs of *test1* and *test2* are so simple that keeping extra information from dialog history would not help. The results of *test1* by the trackers with the LSTMs are even worse than the trackers without the LSTMs since the increased number of parameters raises the problem of over-fitting.

On the other hand, distinct difference between trackers with and without LSTMs is observed in *test3*. Although the accuracies are similar (0.891$\leftrightarrow$0.898), l2 scores show the clear decrease of $0.189 \rightarrow 0.172$ with LSTMs when optimized with two different gradients.

Table 4: Results of the trackers of DSTC1 and our tracker using various models and optimization algorithms, evaluated by the average of all slots. The bold face denotes top score in each evaluation metric.

	Base	1-1	2-2	3-1	4-1	5-1	6-3	7-1	9-4	C	SC	LC	LSC
DSTC1 Test 1													
accuracy	0.712	0.832	0.807	0.808	0.737	0.795	0.867	0.783	0.822	0.819	**0.870**	0.817	0.868
avgp	0.733	0.774	0.771	0.807	0.737	0.787	0.823	0.762	0.794	0.796	**0.841**	0.798	0.837
l2	0.377	0.319	0.322	0.273	0.372	0.300	0.246	0.335	0.290	0.287	**0.223**	0.285	0.229
mrr	0.797	0.875	0.858	0.846	0.813	0.852	0.900	0.843	0.878	0.873	**0.910**	0.870	0.907
roc.v1.eer	0.244	0.126	0.246	0.243	0.737	0.122	**0.118**	0.147	0.143	0.183	0.146	0.173	0.145
roc.v1.05	0.622	0.723	0.672	0.601	0.196	0.710	**0.763**	0.650	0.720	0.677	0.748	0.681	0.744
DSTC1 Test 2													
accuracy	0.546	0.646	0.707	0.683	0.635	0.622	0.790	0.652	0.705	0.834	0.856	0.834	**0.865**
avgp	0.573	0.550	0.629	0.684	0.634	0.615	0.714	0.649	0.651	0.824	**0.846**	0.823	0.842
l2	0.603	0.633	0.503	0.446	0.517	0.535	0.386	0.492	0.476	0.245	**0.213**	0.247	0.220
mrr	0.650	0.717	0.792	0.756	0.713	0.722	0.843	0.744	0.797	0.882	0.900	0.879	**0.904**
roc.v1.eer	0.192	0.197	0.394	0.144	0.635	0.212	0.159	0.189	0.219	0.129	**0.115**	0.124	0.128
roc.v1.05	0.431	0.487	0.516	0.452	0.164	0.480	0.660	0.479	0.490	0.761	**0.793**	0.764	0.786
DSTC1 Test 3													
accuracy	0.789	0.793	0.843	0.819	0.819	0.779	0.835	0.790	0.847	0.819	0.891	0.823	**0.898**
avgp	0.751	0.725	0.757	0.787	0.784	0.701	0.752	0.755	0.740	0.805	0.862	0.809	**0.875**
l2	0.352	0.369	0.323	0.291	0.295	0.395	0.334	0.337	0.343	0.271	0.189	0.266	**0.172**
mrr	0.835	0.851	0.883	0.853	0.853	0.828	0.890	0.841	0.887	0.854	0.919	0.859	**0.924**
roc.v1.eer	0.189	0.164	0.154	0.273	0.124	0.171	0.148	0.119	0.129	0.123	0.101	0.122	**0.095**
roc.v1.05	0.565	0.647	0.681	0.724	0.702	0.623	0.687	0.701	0.738	0.749	0.840	0.752	**0.855**

5.2 DSTC2 Result

Similar to the previous section, we compare our tracker to other trackers submitted to DSTC2. We do not include trackers that additionally use ASR (automatic speech recognition) information in the comparison; we restricted comparison to trackers that only used NLU data as ours for fair comparison.

The following is the summary description provided by the teams:

- Team 1 : linear-chain conditional random field (Kim and Banchs, 2014)
- Team 4 : recurrent neural network (Henderson et al., 2014b)
- Team 6 : maximum-entropy Markov model (Ren et al., 2014)
- Team 7 : combined model of rule-based, maximum-entropy and deep neural network (Sun et al., 2014)

Table 5: Results of the trackers of DSTC2 and our tracker using various models and optimization algorithms, evaluated on the joint slot. The bold face denotes top score in each evaluation metric.

	Base	1-0	3-0	4-3	6-2	7-4	9-0	4 rep	C	SC	LC	LSC
DSTC2 Test												
accuracy	0.719	0.601	0.729	0.737	0.718	0.735	0.499	0.735	0.703	0.728	0.710	**0.741**
avgp	**0.678**	0.503	0.659	0.636	0.638	0.673	0.522	0.641	0.587	0.662	0.605	0.677
l2	0.464	0.649	0.452	0.406	0.437	0.433	0.760	0.402	0.465	0.408	0.441	**0.394**
mrr	0.779	0.661	0.763	0.804	0.772	0.787	0.608	0.801	0.756	0.783	0.780	**0.809**
roc.v1.eer	0.332	**0.096**	0.320	0.461	0.432	0.349	0.313	0.323	0.327	0.352	0.346	0.331
roc.v1.05	0.256	**0.382**	0.249	0.208	0.226	0.243	0.000	0.223	0.245	0.251	0.236	0.285

Evaluation is taken over the joint slot[5], which was the featured metric in DSTC2, that checks the joint correctness of every goal slot. LSTM parameter configuration identical to DSTC1 experiment was also used for this experiment.

Since this is a complex dialog similar to *test3* dataset in DSTC1, the initialization with simple gradient showed a significant performance improvement in both models, with and without LSTMs. On the other hand, LSTM yielded additional significant performance improvement than in DSTC1 datasets, successfully capturing more complex dependencies over dialog turns. While C, SC, LC yielded competitive scores with other teams, LSC scored the highest among the trackers using SLU.

Note that team 4 by Henderson et al. (2014b) adopted a recurrent neural network and results a very similar scores (0.737↔0.741). For a detailed comparison, we replicated the work of team 4 referring to the description in the Henderson's thesis (Henderson, 2015), and presented above as 4 rep. It can be seen that 4 rep is also achieving a very similar score, and the differences in three systems(4-3, 4 rep, and LSC) do not seem significant if the randomness of the learning algorithm is considered.

However, we found out that our system requires much less computational cost when compared to the replicated system. The model itself used ten times smaller number of parameters (3×10^5 parameters used for 4 rep while 3×10^4 parameters used for LSC) to achieve similar or better performance due to the careful choice of component probability models, encoding prior knowledge.

6. Conclusion and Discussion

In this paper, we proposed a robust generative dialog state tracker by using Long-Short Term Memory (LSTM) for the dialog history. Based on the Bayesian filtering in the SDS-POMDP framework, we designed each component probability model to capture important dependencies in dialog state tracking. LSTM is used for user action model, aimed at learning complex system-user action dependencies over time, and has exhibited its performance improvment in complex domains where bookkeeping of important dialog aspects is important for accurate tracking.

We also proposed the two-step optimization algorithm that optimizes the tracker. The performance of a local optimization algorithm used to train the tracker was highly dependent on the initial

5. The performance gain between trackers hence cannot be directly compared with the gains from DSTC1. It is harder to get better score in joint slot than the averaged score on individual slots.

solution, as the objective function is highly nonlinear in the parameter space. We therefore introduced a preliminary optimization stage where we use the gradient descent with approximated gradients calculated by ignoring the dependencies over time steps. The solution from the preliminary optimization was used as the initial solution for the second stage, where we calculated the exact gradients by dynamic programming. This two-step algorithm consistently improved the performance over single-step optimization approach with exact gradient and random initial solution.

We have demonstrated the performance of the tracker and the effectiveness of the optimization algorithm by comparing to the state-of-the-art trackers submitted to Dialog State Tracking Challenges 1 and 2.

Acknowledgments

This work was partly supported by the ICT R&D program of MSIP/IITP [14-824-09-014, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)] and National Research Foundation of Korea (Grant# 2012-007881).

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. A comprehensive reinforcement learning framework for dialogue management optimization. *Selected Topics in Signal Processing, IEEE Journal of*, 6(8):891–902, 2012.

Milica Gasic and Steve Young. Gaussian processes for POMDP-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(1): 28–40, 2014.

Henderson, Blaise Thomson, and Jason Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014a.

Matthew Henderson, Blaise Thomson, and Steve Young. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 467–471, 2013.

Matthew Henderson, Blaise Thomson, and Steve Young. 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)*, page 292, 2014b.

Matthew S Henderson. *Discriminative methods for statistical spoken dialogue systems*. PhD thesis, University of Cambridge, 2015.

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

Daejoong Kim, Jaedeug Choi, Kee-Eung Kim, Jungsu Lee, and Jinho Sohn. Engineering statistical dialog state trackers: a case study on DSTC. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 462–466, 2013.

Seokhwan Kim and Rafael E Banchs. Sequential labeling for tracking dynamic dialog states. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 332, 2014.

Byung-Jun Lee, Woosang Lim, Daejoong Kim, and Kee-Eung Kim. Optimizing generative dialog state tracker via cascading gradient descent. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 273, 2014.

Sungjin Lee. Structured discriminative model for dialog state tracking. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

Sungjin Lee. Extrinsic evaluation of dialog state tracking and predictive metrics for dialog policy optimization. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 310, 2014.

Sungjin Lee and Maxine Eskenazi. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 414–422, 2013.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using Markov Decision Process for Learning Dialogue Strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 201–204, 1998.

Angeliki Metallinou, Dan Bohus, and Jason Williams. Discriminative state tracking for spoken dialog systems. In *Proceedings of the 51st annual meeting of the Association for Computational Linguistics (ACL)*, pages 466–475, 2013.

Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.

Hang Ren, Weiqun Xu, and Yonghong Yan. Markovian discriminative modeling for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 327, 2014.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. The SJTU system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 318, 2014.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2013.

Jason D Williams. Exploiting the ASR n-best by tracking multiple dialog state hypotheses. In *Proceedings of the 2008 INTERSPEECH*, pages 191–194, 2008.

Jason D Williams. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5382–5385, 2010.

Jason D Williams. 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)*, page 282, 2014.

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

Steve Young. Using POMDPs for dialog management. In *Proceeding of 2006 IEEE Workshop on Spoken Language Technology (SLT)*, pages 8–13, 2006.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.