Optimizing Generative Dialog State Tracker via Cascading Gradient Descent

Byung-Jun Lee[1], **Woosang Lim**[1], **Daejoong Kim**[2], **Kee-Eung Kim**[1]
[1] Department of Computer Science, KAIST [2] LG Electronics

Abstract

For robust spoken dialog management, various dialog state tracking methods have been proposed. Although discriminative models are gaining popularity due to their superior performance, generative models based on the Partially Observable Markov Decision Process model still remain attractive since they provide an integrated framework for dialog state tracking and dialog policy optimization. Although a straightforward way to fit a generative model is to independently train the component probability models, we present a gradient descent algorithm that simultaneously train all the component models. We show that the resulting tracker performs competitively with other top-performing trackers that participated in DSTC2.

1 Introduction

Spoken dialog systems, a field rapidly growing with the spread of smart mobile devices, has to deal with challenges to become a primary user interface for natural interaction using conversations. One of the challenges is to maintain the state of the dialog in the conversational process, which is called *dialog state tracking*. The dialog state encapsulates the information needed to successfully finish the dialog, such as users' goal or requests, and thus it is an essential entity in spoken dialog systems. However, the error incurred by Automatic Speech Recognition (ASR) and Spoken Language Understanding (SLU) makes the true user utterance not directly observable, and this makes it difficult to figure out the true dialog state.

Various methods have been used to construct dialog state trackers. The traditional methods used in most commercial systems use hand-crafted rules that typically rely on the most likely result from SLU. However, these rule-based systems are prone to frequent errors as the most likely result is not always correct. Hence, these systems often drive the users to respond using simple keywords and to explicitly confirm everything they say, which is far from a natural conversational interaction. An accurate tracking of the dialog state is crucial for natural and efficient dialogs. On the other hand, modern methods take a statistical approach to calculate the posterior distribution over the dialog states using multiple results from SLU in order to overcome the error in the most likely SLU result.

Statistical dialog state trackers can be categorized into two approaches depending on how the posterior calculation is modeled. The *generative approach* uses the generative model that describes how the SLU results are generated from the hidden dialog state and uses the Bayes' rule to calculate the posterior. It has been a popular approach for statistical dialog state tracking, since it naturally fits into the Partially Observable Markov Decision Process (POMDP) (Williams and Young, 2007), an integrated model for dialog state tracking and dialog strategy optimization. In the POMDP point of view, the dialog state tracking is essentially belief monitoring, which is the task of calculating posterior distribution over the hidden state given the history of observations. Examples of the dialog state trackers that take the generative approach include (Young et al., 2010; Thomson and Young, 2010; Raux and Ma, 2011)

On the other hand, the *discriminative approach* directly models the posterior distribution. Since it avoids modeling of unnecessary aspects of the task, it typically achieves a better tracking accuracy compared to the generative approach. Examples of discriminative dialog state trackers include (Lee, 2013; Metallinou et al., 2013). However, their feature functions often refer to past observations, and it remains yet to be seen whether the discriminative approach can be successfully incorporated into POMDP or reinforcement learning (RL) for dialog strategy optimization.

This paper is concerned with the generative approach to dialog state tracking. In our earlier work (Kim et al., 2013), the optimization of the tracker was carried out independently for each component model (observation model, user action model, and belief refinement model) that comprised our tracker. This was not exactly a proper way to train the tracker for overall performance optimization. In this paper, we present an optimization method, which we call "cascading gradient descent", that trains component models simultaneously. We show that this approach yields a dialog state tracker that performs on par with the best ones that participated in the second Dialog State Tracking Challenge (DSTC2).

The rest of the paper is organized as follows: We briefly review the background of our work in section 2, and present our method in section 3. We then explain the DSTC2 dialog domain and the experimental settings in section 4, and discuss the results in section 5. Finally, we conclude the paper with the summary and the suggestion for future work in section 6.

2 Background and Related Work

The dialog state tracking is formalized as follows: In each turn of the dialog, the spoken dialog system executes system action a, and the user with goal g responds to the system with utterance u. The dialog state in each turn is defined $s = (u, g, h)$, where h is the dialog history encapsulating additional information needed for tracking the dialog state (Williams et al., 2005). The SLU processes the user utterance and generates the results as an N-best list $\boldsymbol{o} = [\langle \tilde{u}_1, f_1\rangle, \dots, \langle \tilde{u}_N, f_N\rangle]$, where $\tilde{u}_i$ is the hypothesized user utterance and f_i is its confidence score[1]. Without loss of generality, we assume that the last item in the N-best list is the null item $\langle \emptyset, 1 - \sum_{i=1}^{N-1} f_i\rangle$, representing the set of unrecognized user utterances. The statistical dialog state tracker maintains the probability distribution over states, called the *belief.*

2.1 Discriminative Dialog State Tracking

Dialog state trackers taking the discriminative approach calculates the belief via trained conditional models that represent the belief directly. *Maximum Entropy* is widely used for the discriminative approach, which formulates the belief as follows:

$$b'(g) = P(g|\boldsymbol{x}) = \eta \cdot exp(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})) \qquad (1)$$

where η is the normalizing constant, $\boldsymbol{x} = (u_1, a_1, g_1, \dots, u_t, a_t, g_t)$ is the history of user actions, system actions, and user goals up to the current dialog turn t, $\boldsymbol{\phi}(\cdot)$ is the vector of feature functions on $\boldsymbol{x}$, and finally, $\boldsymbol{w}$ is the set of model parameters to be learned from dialog data.

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 real dialog domains that have a large number of user goals (the DSTC2 dialog domain used on this paper has 371070 user goals).

Consequently, it is important for the discriminative approach to reduce the size of the state space. (Metallinou et al., 2013) adopts the idea behind the HIS model and confines the set of possible goals to those appeared in SLU results. (Lee, 2013) assumed conditional independence between dialog state components to address scalability, and used conditional random field.

2.2 Generative Dialog State Tracking

In contrast, the generative approach to the dialog state tracking calculates the belief using Bayes' rule, with the belief from the last turn b as a prior and the likelihood given the user utterance hypotheses $\Pr(\boldsymbol{o}|a, g, h)$. In the prior work (Williams et al., 2005), the likelihood is factored and some independence assumptions are made:

$$b'(g', h') = \eta \sum_u \Pr(\boldsymbol{o}|u) \Pr(u|g', a) \cdot \sum_h \Pr(h'|g', u, h, a) \sum_g \Pr(g'|g, a) b(g, h) \qquad (2)$$

where η is the normalizing constant and u is marginalized out in the belief.

Scalability became the important issue, just as in the generative approach. One way to reduce the amount of computation is to group the states into partitions, proposed as the Hidden Information State (HIS) model (Young et al., 2010). Beginning with one root partition with the probability of 1, partitions are split when the distinction is required by observations, i.e. a user utterance

[1]Here we assume that $\sum_{i=1}^{N-1} f_i \leq 1$, acting as a posterior of N-best list.

hypothesis from SLU. This confines the possible goal state to the values that have been appeared as SLU hypotheses, and provides scalability without a loss of accuracy when the coverage of N-best list is large enough to include the true utterance. Using the HIS model with an additional assumption that the user goal does not change (goal transition from one to another is 0), the belief update equation (2) is reformulated as follows:

$$b'(\psi', h') = \eta \sum_u \Pr(\boldsymbol{o}|u) \Pr(u|\psi', a) \cdot \sum_h \Pr(h'|\psi', u, h, a) \Pr(\psi'|\psi) b(\psi, h) \tag{3}$$

where ψ is a set of user goals that share the same belief. Each probability model in the above equation has a name: $\Pr(\boldsymbol{o}|u)$ is called the *observation model*, $\Pr(u|\psi', a)$ is called the *user action model*, $\Pr(\psi'|\psi)$ is called the *belief refinement model*, and $\Pr(h'|\psi', u, h, a)$ is called the *history model*.

In this paper, we used the last turn's belief of dialog states as history state and preserved its dependence in the observation model to improve performance. With the changes, observation model can distinguish user actions without their value. For example, *request alternative* user action may have the power of diminishing dominant partitions, and it can only be learnt by the dependence with partition confidence. The belief update formula used in this paper becomes:

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

Other approaches to cope with the scalability problem in dialog state tracking is to adopt factorized dynamic Bayesian network by making conditional independence assumptions among dialog state components, and use approximate inference algorithms such as loopy belief propagation (Thomson and Young, 2010) or blocked Gibbs sampling (Raux and Ma, 2011).

3 Cascading Gradient Descent

Although equation (4) is an elegant formulation of the dialog state tracking via Bayes rule, there has not been an integrated learning algorithm that simultaneously optimizes component probability models, i.e. the observation, the user action, and the belief refinement models. Our prior work (Kim et al., 2013) relied on independently training each component probability model, and then simply plugging them into (4). Since the independent optimization of component probability models does not lend itself to the optimization of overall dialog state tracking performance, we added an extra post-processing step called "belief transformation" in order to fine tune the results obtained from equation (4). Unfortunately, this effort generally resulted in overfitting to the training data. In this paper, we present an integrated learning algorithm that simultaneously optimizes the component probability models of the HIS model.

We start with defining an objective function which measures the error of the dialog state tracking:

$$E = \sum_{t=1}^{T} \sum_i \frac{1}{2} (b(\psi_i^t) - r_i^t)^2 \tag{5}$$

where t is the dialog turn, i is the partition index, r_i^t is the binary label with value 1 if and only if the partition ψ_i^t contains the true user goal. Note that our objective function coincides with the ℓ_2 performance metrics used in DSTC.

We then express component probability models as functions of features, which are parameterized by sets of weights, and rewrite equation (4):

$$b(\psi_i^t) = \eta^t \sum_{(u^t, f^t) \in \boldsymbol{o}^t} \Pr_{\boldsymbol{w}_\mathrm{O}}(u^t, f^t, a^t, b(\psi_i^{t-1})) \cdot \Pr_{\boldsymbol{w}_\mathrm{U}}(u^t|\psi_i^t, a^t) \Pr_{\boldsymbol{w}_\mathrm{R}}(\psi_i^t|\psi_i^{t-1}) b(\psi_i^{t-1}) \tag{6}$$

where $\boldsymbol{w}_\mathrm{O}$, $\boldsymbol{w}_\mathrm{U}$, and $\boldsymbol{w}_\mathrm{R}$ represent the set of parameters for the observation, the user action, and the belief refinement models, respectively.

Our learning method is basically a gradient descent. The gradient of E with respect to $\boldsymbol{w}_\mathrm{O}$ is derived as follows:

$$\frac{\partial E}{\partial \boldsymbol{w}_\mathrm{O}} = \sum_{t=1}^{T} \sum_i (b(\psi_i^t) - r_i^t) \frac{\partial b(\psi_i^t)}{\partial \boldsymbol{w}_\mathrm{O}}$$

By convenience, we define:

$$\begin{aligned} \delta_i^t = & \sum_{(u^t, f^t) \in \boldsymbol{o}^t} \Pr_{\boldsymbol{w}_\mathrm{O}}(u^t, f^t, a^t, b(\psi_i^{t-1})) \cdot \\ & \Pr_{\boldsymbol{w}_\mathrm{U}}(u^t|\psi_i^t, a^t) \Pr_{\boldsymbol{w}_\mathrm{R}}(\psi_i^t|\psi_i^{t-1}) b(\psi_i^{t-1}) \\ = & \sum_{(u^t, f^t) \in \boldsymbol{o}^t} p_\mathrm{O}^t p_\mathrm{U}^t p_\mathrm{R}^t b(\psi_i^{t-1}) \end{aligned}$$

$$\eta^t = (\sum_i \delta_i^t)^{-1}, \quad b(\psi_i^t) = \eta^t \delta_i^t$$

and then obtain:

$$\begin{aligned}\frac{\partial b(\psi_i^t)}{\partial \boldsymbol{w}_\mathrm{O}} &= \frac{\partial \delta_i^t}{\partial \boldsymbol{w}_\mathrm{O}} \cdot \eta^t + \frac{\partial \eta^t}{\partial \boldsymbol{w}_\mathrm{O}} \cdot \delta_i^t \\ &= \frac{\partial \delta_i^t}{\partial \boldsymbol{w}_\mathrm{O}} \cdot \eta^t - b(\psi_i^t) \sum_{i'} \frac{\partial \delta_{i'}^t}{\partial \boldsymbol{w}_\mathrm{O}} \cdot \eta^t,\end{aligned}$$

where

$$\begin{aligned}\frac{\partial \delta_i^t}{\partial \boldsymbol{w}_\mathrm{O}} =& \Bigg(b(\psi_i^{t-1}) \sum_{(u^t,f^t)\in \boldsymbol{o}^t} \frac{\partial p_\mathrm{O}^t}{\partial \boldsymbol{w}_\mathrm{O}} p_\mathrm{U}^t p_\mathrm{R}^t \\ &+ \frac{\partial b(\psi_i^{t-1})}{\partial \boldsymbol{w}_\mathrm{O}} \sum_{(u^t,f^t)\in \boldsymbol{o}^t} p_\mathrm{O}^t p_\mathrm{U}^t p_\mathrm{R}^t \Bigg).\end{aligned}$$

Gradients for the parameters of other component probability models are derived similarly. We call our algorithm cascading gradient descent since the gradient $\frac{\partial b(\psi_i^t)}{\partial \boldsymbol{w}}$ requires computation of the gradient in the previous dialog turn $\frac{\partial b(\psi_i^{t-1})}{\partial \boldsymbol{w}}$, hence reflecting the temporal impact of the parameter change in throughout the dialog turns.

Once we obtain the gradients, we update the parameters using the gradient descent

$$\begin{aligned}\boldsymbol{w}'_\mathrm{O} &= \boldsymbol{w}_\mathrm{O} - \alpha \left[\frac{\partial E}{\partial \boldsymbol{w}_\mathrm{O}}\right], \\ \boldsymbol{w}'_\mathrm{U} &= \boldsymbol{w}_\mathrm{U} - \alpha \left[\frac{\partial E}{\partial \boldsymbol{w}_\mathrm{U}}\right], \\ \boldsymbol{w}'_\mathrm{R} &= \boldsymbol{w}_\mathrm{R} - \alpha \left[\frac{\partial E}{\partial \boldsymbol{w}_\mathrm{R}}\right]\end{aligned}$$

where α is the stepsize parameter. α is initially set to 0.5 and decreased by multiplying $\frac{1}{10}$ whenever the overall cost function increases.

4 Dialog State Tracking in the Restaurant Information Domain

This section describes the dialog domain used for the evaluation of our dialog tracker and the component probability models used for the domain. An instruction on how to obtain the dataset and a more detailed description on the dialog domain can be found in the DSTC2 summary paper (Henderson et al., 2014).

4.1 Task Description

We used the DSTC2 dialog domain in which the user queries the database of local restaurants. The dataset for the restaurant information domain were originally collected using Amazon Mechanical Turk. A usual dialog proceeds as follows: the user specifies the constraints (e.g. type of food, location, etc) or the name of restaurant he wants, and the system offers the name of a restaurant that qualifies the constraints. User then accepts the offer, and requests for additional information about accepted restaurant. The dialog ends when all the information requested by the user is provided.

The dialog state tracker should thereby clarify three types of information inside the state: goal, method, and requested. The goal state is composed of *name, pricerange, area, and food* slots, which is the information of the constraints that the user has. The method state represents what method user is using to accomplish his goal, whose value is one of the *none, by constraints, by alternatives, by name, or finished.* Lastly, the requested state represents the information currently requested by the user, such as the address, phone number, postal code, etc. In this paper, we restrict ourselves to tracking the goal states only, but our tracker can be easily extended to track others as well.

The dialog state tracker updates the belief turn by turn, receiving SLU N-best hypotheses each with an SLU confidence score in every turn. Despite the large number of states a dialog can have, in the most cases, the coverage of N-best hypotheses is enough to limit the consideration of possible goal state to values that has been observed in SLU hypotheses. Consequently, the task of the dialog state tracker is to generate a set of observed values and their confidence scores for each slot, with the confidence score corresponding to the posterior probability of the goal state being the true goal state. The dialog state tracker also maintains a special goal state, called *None*, which represents that the true goal state has not been observed. Its posterior probability is also computed together with the observed goal states as a part of the belief update. For the rest of this section, we describe the models chosen for each component probabilities.

4.2 Observation Model

The observation model that describes the generation of SLU result for the user utterance is defined as

$$\begin{aligned}&\Pr(\mathbf{o} = \langle u^t, f^t\rangle | u, a, h) = \\ &\qquad \eta_o \Pr_{\boldsymbol{w}_\mathrm{O}}(u^t, f^t, a^t, b(\psi_i^{t-1})) \\ &= \eta_o \frac{1}{1 + \exp(-\boldsymbol{w}_\mathrm{O}^T \boldsymbol{\phi}_\mathrm{O}(u^t, f^t, a^t, b(\psi_i^{t-1})) - b_\mathrm{O})}\end{aligned}$$

user action feature : 34	×	system action feature : 5	×	type of feature : 3	$= 510$
Inform action : 12 [food, pricerange, name, area] × [not_match, slot_match, value_match] % consistency check with system action		offer *or* inform canthelp *or* canthelp.exception		Bias tern (always 1)	
Action with values : 8 [confirm, deny] × [food, pricerange, name, area]	×	expl-conf *or* impl-conf *or* request	×	Value of user confidence f^t	
Action without values : 14 [ack, affirm, bye, hello, negate, repeat, reqmore, reqalts, thankyou, request, null, confirm, deny, inform]		select confirm-domain *or* welcomemsg		Value of last turn's confidence $b(\psi_i^{t-1})$	

Table 1: 510 features used in observation model are specified.

where $\boldsymbol{\phi}_{\mathrm{O}}(u^t, f^t, a^t, b(\psi_i^{t-1}))$ is the vector of features taken from the hypothesized user action u^t, its confidence score f^t generated from SLU, system action a^t, and the belief of partition we are dealing with $b(\psi_i^{t-1})$ from history state. Normalization constante η_o can be ignored since it is subsumed by overall normalization constant η. Feature details are specified in table 1.

4.3 User Action Model

Similar to the observation model, the user action model that predicts the user action given the previous system action and user goal is defined as

$$\begin{aligned}\Pr(u^t|\psi_i^t, a^t) &= \Pr_{\boldsymbol{w}_{\mathrm{U}}}(u^t|\psi_i^t, a^t)\\ &= \frac{exp(\boldsymbol{w}_{\mathrm{U}}^T\boldsymbol{\phi}_{\mathrm{U}}(u^t, \psi_i^t, a^t))}{\sum_u exp(\boldsymbol{w}_{\mathrm{U}}^T\boldsymbol{\phi}_{\mathrm{U}}(u, \psi_i^t, a^t))}\end{aligned}$$

where $\boldsymbol{\phi}_{\mathrm{U}}(u^t, \psi_i^t, a^t) \in \{0,1\}^{322}$ is the vector of features taken from the (hypothesized) user action u^t, system action a^t, and the partition being updated ψ_i^t. Softmax function is used to normalize over possible user actions. Feature details are specified in table 2.

4.4 Belief Refinement Model

The belief refinement model predicts how the partition of the user goal will evolve at the next dialog turn. We defined it as a mixture of the empirical distribution and the uniform distribution obtained from the training data:

$$\begin{aligned}&\Pr_{\boldsymbol{w}_{\mathrm{R}}}(\psi_i^t|\psi_i^{t-1})\\ &= \frac{1}{1+\exp(-\boldsymbol{w}_{\mathrm{R}})}\frac{occurrence(\psi_i^t, \psi_i^{t-1})}{occurrence(\psi_i^{t-1})}\\ &+ \left(1 - \frac{1}{1+\exp(-\boldsymbol{w}_{\mathrm{R}})}\right)\frac{|\psi_i^t|}{|\psi_i^{t-1}|}\end{aligned}$$

where $occurrence(\psi_i^t, \psi_i^{t-1})$ is the number of consecutive dialog turns in the training data with user goals being consistent with ψ_i^{t-1} in the previous turn and ψ_i^t in the current turn, and $occurrence(\psi_i^{t-1})$ is defined similarly for a single turn only. The ratio of the two, which corresponds to the partition split probability used in (Young et al., 2010), is the first term in the mixture. On the other hand, if we use this empirical distribution only, we cannot deal with novel user goals that do not appear in the training data. Assuming that user goals are generated from the uniform distribution, the probability that the user goal is in a partition ψ is $\frac{|\psi|}{N}$ where $|\psi|$ is the number of user goals in the partition ψ, and N is the total number of user goals. The probability that ψ_i^t gets split from ψ_i^{t-1} is then $\frac{|\psi_i^t|}{|\psi_i^{t-1}|}$. Hence, we mix the two probabilities for the resulting model.

The mixture weight is the only parameter of the belief refinement model, which is learned as a part of the cascading gradient descent. Note that we use the sigmoid function in order to make the optimization unconstrained.

user action feature : 35	×	system action feature : 8	+	remaining actions: 42	= 322
Inform action : 24				Confirm/deny action : 16	
[food, pricerange, name, area] × [not_match, slot_match, value_match] % consistency check with system action × [not_match, match] % consistency check with partition	×	[offer *or* inform, canthelp *or* canthelp.exception, expl-conf *or* impl-conf *or* request, select] × [not_match, match] % consistency check with partition	+	[confirm, deny] × [food, pricerange, name, area] × [not_match, match] % consistency check with partition	
Action without values : 11				Remaining system action : 26	
[ack, affirm, bye, hello, negate, repeat, reqalts, reqmore, thankyou, request, null]				[confirm-domain *or* welcomemsg] × [24 inform actions, null, others] % corresponding user actions	

Table 2: 322 features used in user action model are specified.

5 Experimental Details

5.1 Datasets

The restaurant information domain used in DSTC2 is arranged into three datasets: *train, dev, test*. The first two datasets are labeled with the true user goals and user actions to optimize the dialog state tracker before submission. The half of the dialogs are created with artificially degraded speech recognizers, intended to better distinguish the performances of trackers. Details of each dataset are as below:

- **dstc2_train**: Composed of 1612 dialogs of 11405 turns, produced from two different dialog managers with a hand-crafted dialog policy.
- **dstc2_dev**: Composed of 506 dialogs of 3836 turns, produced from the dialog managers used in dstc2_train set. Most of dialog state trackers show lower performance on this dataset than others.
- **dstc2_test**: Composed of 1117 dialogs of 9689 turns, produced from the dialog policy trained by reinforcement learning, which is not used for the train and dev datasets.

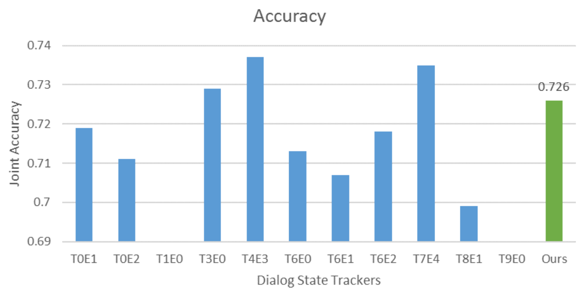
We used both train and dev sets as the training data, as if they were one big dataset. Although the true labels for the test dataset were made public after the challenge, we did not use these labels in any way for optimizing our tracker.

5.2 Pre-training

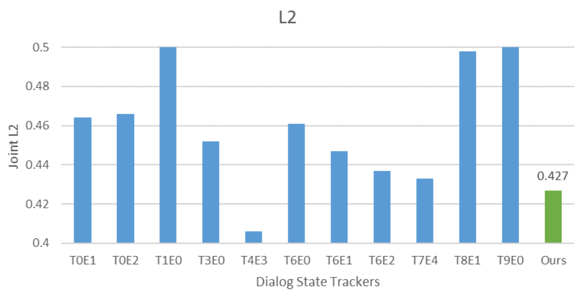
One of the drawbacks in using gradient descent is convergence to a local optimum. We also observed this phenomena during the training of our dialog state tracker via cascading gradient descent. Randomized initialization of parameters is a common practice for gradient descent, but given the high-dimensionality of the parameter space, the randomized initialization had a limited effect in converging to a sufficiently good local optimum.

We adopted a pre-training phase where the parameters of each component model are optimized individually. Once the pre-training is done for each component model, we gathered the parameter values and took them as the initial parameter value for the cascading gradient descent. This pre-training phase helped tremendously converging to a good local optimum, and reduced the number of iterations as well. We pre-trained the parameters of each component model as follows:

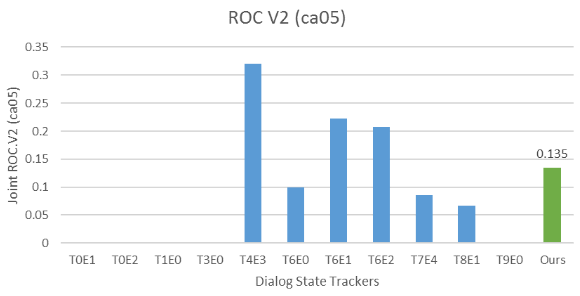
- **Observation Model**: True user action labels in the training set are used as targets for the observation model. For every user action hypothesis in the N-best list, set the target value to 1 if the user action hypothesis is the true user action, and 0 otherwise. A simple gradient descent was used for pre-training.
- **User Action Model**: Although the user action and the system action labels are available, the partition of the user goals is not readily available. However, the latter can be easily obtained by running an unoptimized tracker. Thus, using the labels in the training set and the generated partitions, we set the target value to 1 if the user action hypothesis is the true user action and the partition is consistent with the true user action, and 0 oth-



(a) Evaluation on accuracy metric (higher is better)



(b) Evaluation on L2 metric (lower is better)



(c) Evaluation on $ROC_{V2,ca05}$ metric (higher is better)

Figure 1: The overall results of proposed method. Each figure shows the evaluations over dstc2_test dataset by featured metrics (joint accuracy, joint l2, joint roc.v2) in DSTC2.

erwise. A simple gradient descent was also used for pre-training.

- **Belief Refinement Model**: Since there is only a single parameter for this model, we did not perform pre-training.

5.3 Results and Discussion

Table 3 shows the test set score of tracker implemented based on proposed algorithm, with the score of other trackers submitted to DSTC2. We tried 200 random initialised weights to train model with proposed algorithm, and learned model with the lowest training L2 error is picked to show the result on the test set. Because we only used live SLU and past data to track dialog state, other tracker results with the same condition are selected to compare with our tracker.

The implementation of our algorithm was not ready until the DSTC2 deadline. We participated as Team 8 using the old optimization method in (Kim et al., 2013). As shown in the table 3, the new algorithm shows a substantial improvement, achieving almost 15% decrease in the L2 error. Since both trackers are fine-tuned, this improvement seems to be originated from the new optimization algorithm.

For all three featured metrics used to evaluate, tracker constructed with our proposed method shows competitive performance. The key to excel baseline tracker was to discover the relation between user action and system action. For example, user actions that tell about the same slot system was talking about but giving different value are usually correcting wrong recognitions so far, which should significantly reduce the belief over state the system was tracking.

Due to the objective function that is designed to optimize L2 error, our tracker shows better performance at L2 error than the other metrics. For both all goal metric and joint goal metric, our tracker shows low L2 error when compared to other trackers while the rank of accuracy metric is not so high. When the fact that our method as a generative state tracker benefits from the ability to be easily incorporated into POMDP framework is considered, only similar performance to other trackers is satisfactory.

6 Conclusion

In this paper, we propose a simple method that optimizes overall parameters of generative state tracker using ”Cascading Gradient Descent” al-

All goal

Team	0		1	3	4	6			7	8	9	Ours
Entry	1	2	0	0	3	0	1	2	4	1	0	
Accuracy	0.886	0.88	0.837	0.892	0.895	0.884	0.882	0.885	0.894	0.873	0.77	0.886
AvgP	0.865	0.852	0.778	0.856	0.853	0.789	0.833	0.843	0.862	0.827	0.782	0.846
L2	0.192	0.198	0.289	0.189	0.17	0.197	0.189	0.184	0.179	0.227	0.358	0.186
MRR	0.918	0.914	0.87	0.911	0.927	0.917	0.916	0.918	0.922	0.904	0.833	0.918
ROC$_{V1,ca05}$	0.777	0.767	0.0	0.778	0.842	0.773	0.786	0.809	0.806	0.635	0.0	0.805
ROC$_{V1,eer}$	0.139	0.133	0.0	0.119	0.103	0.135	0.123	0.116	0.116	0.163	0.219	0.120
ROC$_{V2,ca05}$	0.0	0.0	0.0	0.0	0.3	0.27	0.417	0.384	0.154	0.0	0.0	0.197
UpdateAcc	0.886	0.881	0.837	0.891	0.895	0.882	0.88	0.883	0.894	0.873	0.769	0.886
UpdatePrec	0.898	0.897	0.846	0.904	0.907	0.898	0.895	0.897	0.903	0.886	0.804	0.896

Table 3: Test set scores averaged over all goal slots of our proposed algorithm and other trackers are presented. The goal slots are composed of food, pricerange, name and area.

Joint goal

Team	0		1	3	4	6			7	8	9	Ours
Entry	1	2	0	0	3	0	1	2	4	1	0	
Accuracy	0.719	0.711	0.601	0.729	0.737	0.713	0.707	0.718	0.735	0.699	0.499	0.726
AvgP	0.678	0.66	0.503	0.659	0.636	0.54	0.619	0.638	0.673	0.583	0.522	0.658
L2	0.464	0.466	0.649	0.452	0.406	0.461	0.447	0.437	0.433	0.498	0.76	0.427
MRR	0.779	0.757	0.661	0.763	0.804	0.767	0.765	0.772	0.787	0.749	0.608	0.775
ROC$_{V1,ca05}$	0.332	0.316	0.096	0.32	0.461	0.324	0.395	0.432	0.349	0.22	0.0	0.438
ROC$_{V1,eer}$	0.256	0.254	0.382	0.249	0.208	0.281	0.241	0.226	0.243	0.299	0.313	0.218
ROC$_{V2,ca05}$	0.0	0.0	0.064	0.0	0.321	0.1	0.223	0.207	0.086	0.067	0.0	0.135
UpdateAcc	0.489	0.487	0.37	0.495	0.507	0.473	0.466	0.476	0.514	0.459	0.325	0.488
UpdatePrec	0.729	0.694	0.677	0.759	0.726	0.748	0.743	0.743	0.703	0.692	0.54	0.71

Table 4: Test set scores of joint goal slot of our proposed algorithm and other trackers are presented. The joint goal slot is a slot that is treated as correct when every goal slot is correct.

gorithm. Using proposed method on Hidden Information State model, we construct a tracker that performs competitively with DSTC2 participants, who mostly adopt discriminative approaches. Since generative approach has much more potential to be extended to more complex models or toward different domains such as DSTC3, our tracker has the advantage over the other trackers.

Hidden Information State (HIS) model with cascading gradient descent has far more steps of improvement remaining. Although history state in current paper only includes previous partition belief due to implementation convenience, utilizing additional history state is the key to improve performance even more. History state can include any information depending on how we define the state. The reason why the discriminative state tracking methods generally show good performance in terms of accuracy is rich set of potentially informative features, which can be employed by the history state.

In addition to the future improvements with history state, we can consider improving each probability models. In this paper, probability models are modeled with sigmoid function or softmax function over weighted features, which is in other words a neural network with no hidden layer. The model used in this paper can naturally developed by adding hidden layers, and ultimately deep learning techniques could be applicable. Applying deep learning techniques could help the history state to find out influential hidden features to employ.

Acknowledgments

This work was supported by the IT R&D program of MKE/KEIT. [10041678, The Original Technology Development of Interactive Intelligent Personal Assistant Software for the Information Service on multiple domains]

References

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the SIGdial 2014 Conference*, Baltimore, U.S.A., June.

Daejoong Kim, Jaedeug Choi, Kee-Eung Kim, Jungsu Lee, and Jinho Sohn. 2013. A specific analysis of a dialog state tracker in a challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 462–466.

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. *Proceedings of the SIGDIAL 2013 Conference*, pages 442–451.

Angeliki Metallinou, Dan Bohus, and Jason D Williams. 2013. Discriminative state tracking for spoken dialog systems. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguastics*, pages 466–475.

Antoine Raux and Yi Ma. 2011. Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems. In *INTERSPEECH*, pages 801–804.

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.

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

Jason D Williams, Pascal Poupart, and Steve Young. 2005. Factored partially observable markov decision processes for dialogue management. In *4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, pages 76–82.

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