THE USE OF DISCRIMINATIVE BELIEF TRACKING IN POMDP-BASED DIALOGUE SYSTEMS

Dongho Kim, Matthew Henderson, Milica Gašić, Pirros Tsiakoulis, Steve Young

Department of Engineering, University of Cambridge, Cambridge, UK

{dk449,mh521,mg436,pt344,sjy}@cam.ac.uk

ABSTRACT

Statistical spoken dialogue systems based on Partially Observable Markov Decision Processes (POMDPs) have been shown to be more robust to speech recognition errors by maintaining a belief distribution over multiple dialogue states and making policy decisions based on the entire distribution rather than the single most likely hypothesis. To date most POMDP-based systems have used generative trackers. However, concerns about modelling accuracy have created interest in discriminative methods, and recent results from the second Dialog State Tracking Challenge (DSTC2) have shown that discriminative trackers can significantly outperform generative models in terms of tracking accuracy. The aim of this paper is to investigate the extent to which these improvements translate into improved task completion rates when incorporated into a spoken dialogue system. To do this, the Recurrent Neural Network (RNN) tracker described by Henderson et al in DSTC2 was integrated into the Cambridge statistical dialogue system and compared with the existing generative Bayesian network tracker. Using a Gaussian Process (GP) based policy, the experimental results indicate that the system using the RNN tracker performs significantly better than the system with the original Bayesian network tracker.

Index Terms— dialogue management, spoken dialogue systems, recurrent neural networks, belief tracking, POMDP

1. INTRODUCTION

Recent advances in statistical POMDP-based spoken dialogue systems have demonstrated increased robustness to speech recognition errors[1]. A key aspect of this approach is that a belief distribution over multiple states is maintained, allowing the dialogue policy to make decisions based on the entire distribution rather than the single most likely hypothesis.

To date, most POMDP-based dialogue systems have used generative belief trackers. However, it has been shown that incorrect modelling assumptions and inaccurately estimated parameters cause performance to degrade [2]. In particular, most current generative trackers do not accurately model correlations in Automatic Speech Recognition (ASR) errors resulting in the potential to overestimate the probabilities of ASR hypotheses which were actually spurious artefacts of the way the recogniser works. Discriminative belief trackers provide an attractive alternative and recent results from the second Dialog State Tracking Challenge (DSTC2) have shown that discriminative trackers can significantly outperform generative models in terms of tracking accuracy.[1]

The aim of this research is to investigate the extent to which these improvements in belief tracking translate into improved system performance, measured by average rewards and task completion rates, when incorporated into a spoken dialogue system. To do this, the Recurrent Neural Network (RNN) tracker [3] has been integrated into the Cambridge statistical dialogue system and compared with the existing generative Bayesian Network (BN) tracker.

The remainder of the paper is structured as follows. Section 2 reviews the discriminative RNN tracker used in [3] and the changes needed to integrate it into a full dialogue system. Section 3 describes the Cambridge statistical dialogue system and the experimental set-up. Section 4 then provides comparative results for belief tracking and dialogue performance using the RNN and BN trackers.

2. RNN BELIEF TRACKER

RNNs provide a natural model for belief tracking in dialogue, as they are able to model and and classify dynamic sequences with complex behaviours from step to step. The role of a POMDP belief tracker is to implement the belief update equation $\mathbf{b}' = \tau(\mathbf{b}, a, o)$ where $\mathbf{b}$ is the belief, a is the agent's last action, o is the current observation, and $\mathbf{b}'$ is the updated belief. An RNN tracker therefore takes as input the last observation from the user and the last system action, updates its internal memory and outputs an updated belief. The observation is typically the output of a Spoken Language Understanding (SLU) component in the form of a semantic decoder, which

This work was supported by the EC FP7 programme FP7/2011-14 under grant agreement no. 287615 (PARLANCE).

[1] The term *dialog state tracking* normally refers to tracking the most likely hypothesis while *belief tracking* refers to tracking the full probability distribution. We use these terms interchangeably in this paper since all the trackers we consider output the full belief distribution.

maps the ASR hypotheses to an N-best list of semantic hypotheses.

2.1. Belief Tracking Problem Definition

The domain of the belief tracker considered in this paper is a modified version of the second Dialog State Tracking Challenge (DSTC2) which consists of a large corpus of telephone-based dialogues in the restaurant domain [4, 3] produced by subjects recruited using Amazon Mechanical Turk. Each subject was asked to find a restaurant in Cambridge by specifying values for upto three goal *slots*: *area*, *food*, and *pricerange*; and obtain information about the selected venue from the system. DSTC2 belief trackers must therefore output belief distributions at every turn for the user goals, the retrieval method, and the requested information slots. In order to integrate a tracker into the Cambridge dialogue system an additional DiscourseAct must be determined. Therefore, the belief state of the extended DSTC2 tracker has the following components:

Goal A probability distribution over the user's goal for each slot. Each distribution ranges over all possible values for that slot, plus the value *None*, to indicate that a valid value has not been mentioned yet. In the experiments, the joint goal is reported as a product of the marginal distributions per slot.

Method A probability distribution over methods, which encodes how the user is trying to retrieve information from the system. For example, *byconstraints*, when the user is trying to constrain the search by specifying slot values, and *finished*, when the user wants to end the dialogue.

Requested slots The probability that each informational slot (e.g. address, telephone number, ...) has been requested by the user. The slots requested by the user are labeled as *requested*.

DiscourseAct A probability distribution over possible discourse actions. Examples include saying hello, asking for the system to repeat, and saying thank you.

In passing, it should also be noted that some corrections were required to the DSTC2 annotations. For example, a slot should be labeled as *requested* when the user performs a *request(slot)* action, so that the value of the slot can be subsequently informed by the system. Whereas directly requested slots were annotated, user confirmation requests *confirm(slot=value)* were not annotated and had to be added. Full details of the challenge are given in [4, 3].

2.2. Feature Representation

The features used in [3] consist of n-grams at both the ASR word level and the SLU dialogue act level. Since evaluation and training of a full dialog system require the use of a simulator operating at the semantic level, the RNN tracker studied in this paper uses only n-gram features at the dialogue act level.

Dialogue acts in the Cambridge restaurant domain consist of a list of the form *acttype(slot=value)[p]* where p is the probability of the act and the *slot=value* pair is optional. The n-gram type features extracted from each such component act are *'acttype'*, *'slot'*, *'value'*, *'acttype slot'*, *'slot value'* and *'acttype slot value'*, or just *'acttype'* for the dialogue act *acttype()*. Since the user action is an N-best list of dialogue acts, the n-gram features are weighted by the probabilities of each act and summed to give a single vector. The system action has a probability of 1, so it is encoded in the same way except that each feature is given weight 1.

The RNN tracker which takes an SLU N-best list and the last system action as input at each turn is shown in Table 1. The features from the SLU N-best list and the last system actions are combined to form the vector $\mathbf{f}$ in Table 1.

In order to make the RNN tracker more generalisable, delexicalised n-gram features are also incorporated[3]. The key idea is to replace all occurrences of a particular slot or slot value with a tag like '⟨*slot*⟩' or '⟨*value*⟩'. Table 1 shows the process of creating the tagged feature vectors, $\mathbf{f}_s$ and $\mathbf{f}_v$ from the untagged feature $\mathbf{f}$.

	SLU		System Act	
	inform(food=jamaican)	0.9	confirm(food=jamaican)	
	inform(food=indian)	0.1		
$\mathbf{f}$	inform food	1.0	confirm food	1.0
	inform food jamaican	0.9	confirm food jamaican	1.0
	inform food indian	0.1	food jamaican	1.0
	food jamaican	0.9	confirm	1.0
	food indian	0.1	food	1.0
	inform	1.0	jamaican	1.0
	food	1.0		
	jamaican	0.9		
	indian	0.1		
$\mathbf{f}_s$	inform ⟨slot⟩ ⟨value⟩	1.0	confirm ⟨slot⟩ ⟨value⟩	1.0
	⟨slot⟩ ⟨value⟩	1.0	⟨slot⟩ ⟨value⟩	1.0
	⟨slot⟩	1.0	⟨slot⟩	1.0
	⟨value⟩	1.0	⟨value⟩	1.0
	inform food ⟨value⟩	1.0	confirm food ⟨value⟩	1.0
	food ⟨value⟩	1.0	food ⟨value⟩	1.0
	inform ⟨slot⟩ jamaican	0.9	confirm ⟨slot⟩ jamaican	1.0
	inform ⟨slot⟩ indian	0.1	⟨slot⟩ jamaican	1.0
	⟨slot⟩ jamaican	0.9		
	⟨slot⟩ indian	0.1		
$\mathbf{f}_v$	inform food ⟨value⟩	0.9	confirm food ⟨value⟩	1.0
	food ⟨value⟩	0.9	food ⟨value⟩	1.0
	⟨value⟩	0.9	⟨value⟩	1.0

Table 1: Example of feature extraction for one turn, showing $\mathbf{f}$, $\mathbf{f}_{s=\text{food}}$ and $\mathbf{f}_{v=\text{jamaican}}$. For all $v \notin \{\text{indian}, \text{jamaican}\}$, $\mathbf{f}_v = \mathbf{0}$. This is the SLU version of Figure 1 in [3].

2.3. RNN Model

This section briefly explains the RNN structure. In what follows, the notation $\oplus$ denotes vector concatenation and NNet$(\cdot)$ denotes a neural network function of the input. In this paper all networks have one hidden layer with a sigmoidal activation function. Each RNN also holds an internal memory, $\mathbf{m}$, which is updated at each step. The output belief $\mathbf{b}$ is the probability distribution over possible values as described in Section 2.1. Therefore in one turn $\mathbf{b}$ and $\mathbf{m}$ are updated to give the new belief $\mathbf{b}'$ and memory $\mathbf{m}'$.

The RNN for tracking the goal for a given slot s consists of two parts. The first part uses untagged features to generate output vector $\mathbf{h}$, which is obtained from the untagged inputs as follows:

$$\mathbf{h} = \text{NNet}(\mathbf{f} \oplus \mathbf{b} \oplus \mathbf{m})$$

where $\mathbf{f}$ are the untagged features, $\mathbf{b}$ is the previous belief vector and $\mathbf{m}$ is the hidden memory.

The network for $\mathbf{h}$ requires examples of every value in training, and might be prone to poor generalisation. For each value v, a component is calculated using a neural network which additionally takes tagged features $\mathbf{f}_s$ and $\mathbf{f}_v$ as input:

$$\mathbf{g}_v = \text{NNet}\left(\mathbf{f} \oplus \mathbf{f}_s \oplus \mathbf{f}_v \oplus \{\mathbf{b}_v, \mathbf{b}_{None}\} \oplus \mathbf{m}\right) \in \mathbb{R}.$$

The components $\mathbf{g}_v$ are then combined to form a second vector $\mathbf{g}$. By using regularisation, the learning will prefer where possible to use the network for $\mathbf{g}$ rather than learning the individual weights for each value required in the network for $\mathbf{h}$. This network is able to deal with unseen or infrequently seen dialogue states, so long as the state can be tagged in the feature extraction. This model can also be shared across slots since $\mathbf{f}_s$ is included as an input.

The above networks applied to tagged and untagged inputs are combined to give the new belief $\mathbf{b}'$:

$$\mathbf{b}' = \text{softmax}([\mathbf{h} + \mathbf{g}] \oplus \{B\})$$

where B is a parameter of the RNN. The contribution from $\mathbf{g}$ may be seen as accounting for the general behaviour of tagged hypotheses, while $\mathbf{h}$ makes corrections due to correlations with untagged features and value specific behaviour.

Finally, the memory is updated according to the logistic regression:

$$\mathbf{m}' = \sigma(W_{m_0}\mathbf{f} + W_{m_1}\mathbf{m})$$

where the W_{m_i} are parameters of the RNN.

A similar RNN is used to track the requested slots. Here v runs over all the requestable slots, and requestable slot names are tagged in the feature vectors $\mathbf{f}_v$. This allows the neural network calculating $\mathbf{g}$ to learn general patterns across slots just as in the case of goals. The equation for $\mathbf{b}'$ is changed to:

$$\mathbf{b}' = \sigma(\mathbf{h} + \mathbf{g})$$

so each component of $\mathbf{b}'$ represents the probability of a slot being requested. For method and discourseAct classification, the same RNN structure as for a goal is used. No tagging of the feature vectors is used in this case.

3. THE DIALOGUE SYSTEM

The Cambridge statistical dialogue system is a domain independent system, which is trainable from data by modelling the dialogue as a POMDP. In the POMDP framework, the system observes an action from the user, updates the belief state and automatically chooses the system action at every time step. The system is optimised to maximise rewards over the dialogue, by training its policy which can be defined as a mapping from belief to action. Hence, the accuracy of belief trackers is crucial for learning how to manage dialogues successfully.

For the experiments in this paper, the system is configured for the restaurant domain with a Gaussian Process (GP) policy [5] and two alternative belief trackers: a generative BN tracker [6] and the discriminative RNN tracker described in Section 2. The restaurant domain consists of about 150 restaurants in Cambridge, UK that were automatically extracted from the TopTable web service.

In the following subsections, the user simulator and error model, which are responsible for simulating realistic spoken dialogues on semantic level will be explained for completeness. Note that the generated dialogues are used as training and test data for belief trackers as well as the GP policy. The method for policy optimisation is reviewed in the last subsection.

3.1. Agenda-Based User Simulator and Error Model

Since policy training demands a large amount of data, large-scale comparative studies are greatly facilitated by the use of a user simulator. In this paper, we used an agenda-based user simulator developed by originally by Schatzmann [7, 8] which factorises the user state into an agenda and a goal. The goal ensures that the user simulator exhibits consistent, goal-directed behaviour. The role of the agenda is to elicit the dialogue acts that are needed for the simulated user to fulfil the required goal. Both the goal and the agenda are dynamically updated throughout the dialogue. The updates are sometimes stochastic to enable a wide spread of realistic dialogues to be generated.

The simulator includes an error model which adds confusions to the simulated user action such that it resembles those found in real data [9]. In the experiments below, the maximum length of the N-best list output at each turn by the simulator was set to 3. In order to test robustness to differing amounts of speech understanding errors, the confusion rate was set to 15%, 30% or 45%, which indicates the probability that the true hypothesis is not included in the N-best list.

3.2. Policy optimisation

GP-based reinforcement learning has been recently applied to POMDP dialogue policy optimisation in order to exploit the correlations between different belief states and thus speed up the learning process [5]. In the GP-Sarsa algorithm [5], the Q-function, which is a mapping of a belief-action pair to its expected cumulative reward, is non-parametrically modelled as a Gaussian process. This GP-based Q-function therefore defines a Gaussian distribution for every belief-action pair. During the training process, the algorithm iteratively samples Q-values from the Gaussian distribution for each action and selects the best system action. After observing each immediate reward, the Q-function is updated based on this experience. In practice the variance of the Q-function is usually scaled by a factor η to mitigate inaccurate estimation of variances.

In the existing system using the BN tracker, some domain knowledge is used in a form of constraints on system actions. For example, the system retains a list of the slot values which may have been mentioned by the user because they have appeared in the SLU N-best hypotheses. The *inform(slot=value)* system action, which tells the user what the system believes the value of the slot to be, is not executable when no value has mentioned by the user. Similarly, when there are less than two values in the list, the system cannot perform the *select(slot=value1,slot=value2)* action, which requests the user to choose between two values. The use of these constraints is reasonable in the BN tracker since SLU hypotheses are directly used as input and the probabilities for unobserved values are effectively zero. However, there is no notion of previously mentioned values in the RNN tracker because the SLU hypotheses are used as features and the correlation between slot values can generate significant probability for unobserved values. For fair comparison, therefore, we trained and tested the BN-based system without those constraints. The result of the BN tracker with action constraints is also reported as reference. Without action constraints, there are more possible actions to choose at each turn. In order to encourage exploration during policy training, the variance scale factor η was set to 9, which is a factor of three higher than normal. All the Gaussian processes were configured to use Gaussian kernels.

Finally, the reward function was set to give a reward of 20 for successful dialogues, zero otherwise. For each dialogue turn, a reward of 1 is deducted to encourage shorter dialogues, and an additional 4 is deducted when the system offers a venue which does not satisfy the user's goal. The discount factor γ was set to 1.

4. RESULTS

4.1. Tracking Results

To test belief tracking accuracy, 5000 dialogues were generated using the agenda-based simulated user and a handcrafted policy. A randomly chosen set of 4500 dialogues was used to train the RNN tracker and the remaining 500 dialogues were used for testing. The RNN tracker was compared to the existing generative BN tracker and the DSTC2 focus-based baseline tracker [10]. The RNN was initialised before training, using the denoising autoencoder [11] and the shared initialisation method [12].

For the evaluation the conventions prescribed in DSTC2 were used, in which turns are only evaluated when there is some information about the state component in the dialogue so far. Note that the true state label is accumulated forwards through the dialogues. For example the goal for slot s is *None* until it is informed as $s = v$ by the user, at which point it becomes v until it is informed otherwise. The performance was measured using two metrics, accuracy and L2 distance. The accuracy indicates the fraction of turns in which the tracker's 1-best hypothesis is correct, and the L2 distance is calculated between the belief distribution output by the tracker and the delta distribution which has a probability of 1 on the label.

As shown in Table 2, in terms of accuracy and L2 distance, the RNN tracker outperformed the other trackers for all dialogue state components and error rates. The BN tracker provided only small improvements compared to the baseline. For all trackers, the performance was degraded at the higher error rate as expected. However, even with 45% error rate, the RNN tracker achieved 84.6% accuracy for Joint Goals while the BN tracker achieved only 50.7% accuracy, and or for the Method, Requested Slots and DiscourseAct, the RNN tracker maintained around 97% accuracy.

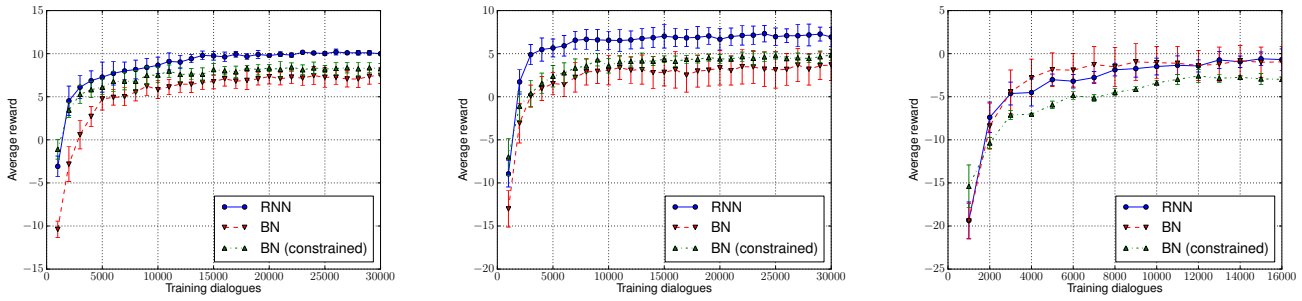
4.2. Dialogue Management Result

In order to investigate the effects of belief tracker accuracy on the overall system performance, GP policies were trained using the BN and RNN trackers on the simulated dialogues. Results during training with up to 30000 dialogues are given in Figure 1 and Figure 2. Note that each point in the plots represents the performance, gathered during one training session on 1000 dialogues, with exploration. To avoid variability in GP policy training, 5 policies were trained for each setup. The error bar indicates the 95% confidence interval.

As shown in Figure 1a and Figure 1b, it is clear that the better accuracy of the RNN tracker introduced a significant improvement in system performance. The GP policies with the RNN tracker showed significantly higher average reward at error rates of 15% and 30%, and obtained similar (Figure 2b) or higher (Figure 2a) success rates. However, at the 45% error rate, the improvement in RNN tracking performance was not translated into the better performance of the

	Joint Goals		Method		Requested Slots		DiscourseAct	
error=15%	Acc	L2	Acc	L2	Acc	L2	Acc	L2
RNN	**0.937**	**0.098**	**0.984**	**0.025**	**0.970**	**0.046**	**0.988**	**0.018**
BN	0.644	0.506	0.884	0.213	0.876	0.275	0.962	0.055
Baseline	0.670	0.456	0.864	0.233	0.832	0.340	0.971	0.060
error=30%	Acc	L2	Acc	L2	Acc	L2	Acc	L2
RNN	**0.935**	**0.108**	**0.978**	**0.036**	**0.973**	**0.044**	**0.985**	**0.022**
BN	0.688	0.480	0.897	0.188	0.871	0.307	0.961	0.060
Baseline	0.670	0.443	0.863	0.237	0.847	0.336	0.968	0.063
error=45%	Acc	L2	Acc	L2	Acc	L2	Acc	L2
RNN	**0.846**	**0.244**	**0.971**	**0.045**	**0.966**	**0.051**	**0.976**	**0.037**
BN	0.507	0.667	0.888	0.207	0.851	0.304	0.945	0.083
Baseline	0.507	0.606	0.843	0.270	0.797	0.406	0.952	0.088

Table 2: Belief tracking performance. Acc denotes the accuracy of the most likely belief at each turn, and L2 denotes the squared L2 distance between the estimated belief distribution and correct delta distribution.



(a) 15% error rate (b) 30% error rate (c) 45% error rate

Fig. 1: Average reward during GP policy training with $\eta = 9$.

policy, as shown in Table 2 and Figure 1c. We suspect this is due to the discriminative model which is trained to maximise accuracy of the best hypothesis not the distribution across possible hypotheses, and thus might be overconfident despite the very high error rate.

The BN tracker shows poor tracking performance, especially when the user frequently changes his goal during the dialogue. To highlight the advantage of the RNN tracker, we tested the policies, trained with 30000 dialogues, on 1000 dialogues in which the user changes his mind at least once. For the test policy, η is set to 1 and exploration is disabled. Table 3 compares the results on normal dialogues to those on the dialogues with frequent goal changes. The RNN results suggest that performance degradation on goal changes is less than the case of the BN tracker.

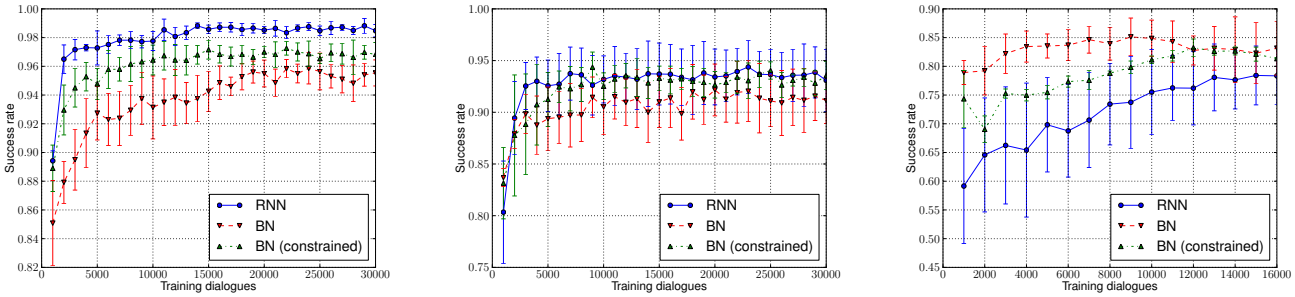
	Normal		Goal change	
error=15%	Reward	Success	Reward	Success
RNN	**10.389**	**0.976**	**7.979**	**0.976**
	±0.557	±0.009	±0.502	±0.009
BN	6.064	0.907	0.072	0.850
	±0.757	±0.018	±0.925	±0.022
error=30%	Reward	Success	Reward	Success
RNN	**7.256**	**0.948**	**1.861**	**0.937**
	±0.674	±0.014	±0.971	±0.015
BN	0.466	0.798	−6.742	0.736
	±0.936	±0.025	±1.078	±0.027
error=45%	Reward	Success	Reward	Success
RNN	**−0.645**	**0.727**	**−6.920**	**0.651**
	±1.005	±0.028	±1.203	±0.030
BN	TBA	TBA	TBA	TBA
	TBA	TBA	TBA	TBA

Table 3: Performance on normal dialogues and with frequent goal changes.

5. CONCLUSIONS

In this paper, a discriminative recurrent neural network (RNN) based belief tracker has been integrated into a spoken dialogue system in order to compare performance with an existing generative Bayes network (BN) based tracker. The results confirm earlier findings that the RNN tracker delivers significantly higher tracking accuracy than the BN tracker over a range of error rates. Furthermore, at error rates of 30% or less, this improved accuracy leads to significant improvements in task success rates in a complete dialogue system.

A key difference between the BN and RNN approaches is that the BN tracker only assigns significant probability mass



(a) 15% error rate (b) 30% error rate (c) 45% error rate

Fig. 2: Success rate during GP policy training with $\eta = 9$.

to values that have actually appeared in the decoded user inputs, whereas the RNN can assign mass to any part of the output distribution. At very high error rates, belief space becomes very noisy in the RNN case, and the GP-based policy learning becomes very slow. As a consequence, the discriminative tracker does not perform as well. This suggests the need to investigate other models for predicting target distributions such as mixture density RNNs[13].

Having established, the potential of discriminative RNN belief trackers, the next step will be to train and test a complete end-to-end POMDP-based dialogue system via human interaction [14]. This will also enable word-based RNN trackers [3] mapping directly from ASR results to belief without using an explicit semantic decoder to be explored.

6. REFERENCES

[1] SJ Young, C Breslin, M Gasic, M Henderson, D Kim, M Szummer, B Thomson, P Tsiakoulis, and E Tzirkel Hancock, "Evaluation of statistical pomdp-based dialogue systems in noisy environments," in *Int Workshop Spoken Dialogue Systems*, 2014.

[2] Jason D. Williams, "A critical analysis of two statistical spoken dialog systems in public use," in *Proc. of SLT*, Dec. 2012, pp. 55–60.

[3] Matthew Henderson, Blaise Thomson, and Steve Young, "Word-based dialog state tracking with recurrent neural networks," in *Proc. of SIGdial*, 2014.

[4] Matthew Henderson, Blaise Thomson, and Jason Williams, "Dialog State Tracking Challenge 2 & 3 Handbook," `http://camdial.org/~mh521/dstc/downloads/handbook.pdf`, 2013.

[5] Milica Gašić and Steve Young, "Gaussian processes for POMDP-based dialogue manager optimisation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 28–40, 2014.

[6] Blaise Thomson and Steve Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech & Language*, vol. 24, no. 4, pp. 562–588, Oct. 2010.

[7] Jost Schatzmann, *Statistical user and error modelling for spoken dialogue systems*, Ph.D. thesis, University of Cambridge, 2008.

[8] Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young, "Parameter estimation for agenda-based user simulation," *Proc. of SIGdial*, pp. 116–123, 2010.

[9] Blaise Thomson, Milica Gašić, Matthew Henderson, Pirros Tsiakoulis, and Steve Young, "N-best error simulation for training spoken dialogue systems," in *Proc. of SLT*, 2012.

[10] Matthew Henderson, Blaise Thomson, and Jason Williams, "The second dialog state tracking challenge," in *Proc. of SIGdial*, 2014.

[11] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. of the 25th Int. Conf. on Machine Learning*, 2008.

[12] Matthew Henderson, Blaise Thomson, and Steve Young, "Deep neural network approach for the dialog state tracking challenge," in *Proc. of SIGdial*, 2013.

[13] CM Bishop, "Mixutre density networks," Report NCRG/94/004, Neural Computing Research Group, Aston University, 1994.

[14] Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young, "On-line policy optimisation of Bayesian spoken dialogue systems via human interaction," in *Proc. of ICASSP*, 2013.