ROBUST DIALOG STATE TRACKING USING DELEXICALISED RECURRENT NEURAL NETWORKS AND UNSUPERVISED ADAPTATION

Matthew Henderson[1]*, Blaise Thomson*[2] *and Steve Young*[1]

[1]Department of Engineering, University of Cambridge, UK
[2]VocalIQ Ltd., Cambridge, UK

mh521@eng.cam.ac.uk blaise@vocaliq.com sjy@eng.cam.ac.uk

ABSTRACT

Tracking the user's intention throughout the course of a dialog, called dialog state tracking, is an important component of any dialog system. Most existing spoken dialog systems are designed to work in a static, well-defined domain, and are not well suited to tasks in which the domain may change or be extended over time. This paper shows how recurrent neural networks can be effectively applied to tracking in an extended domain with new slots and values not present in training data. The method is evaluated in the third Dialog State Tracking Challenge, where it significantly outperforms other approaches in the task of tracking the user's goal. A method for online unsupervised adaptation to new domains is also presented. Unsupervised adaptation is shown to be helpful in improving word-based recurrent neural networks, which work directly from the speech recognition results. Word-based dialog state tracking is attractive as it does not require engineering a spoken language understanding system for use in the new domain and it avoids the need for a general purpose intermediate semantic representation.

***Index Terms*—** Dialog state tracking, dialog systems, spoken language understanding.

1. INTRODUCTION

Spoken dialog systems allow a user to achieve a task, such as finding a restaurant, using natural language. Before being deployed, most dialog systems are trained for well-defined and static domains such as this. However applications where the domain may grow and change over time are of particular interest, especially considering the desire for machines which can speak on any topic at web-scale. One key issue in allowing for expanding domains is designing systems which can estimate the *state* of the conversation, as the set of possible states grows with the domain.

This paper presents how recurrent neural networks (RNNs) can be effectively applied to the task of dialog state tracking in expanding domains. Dialog state tracking has been brought into focus recently with the first and second Dialog State Tracking Challenges (DSTCs) [1, 2]. The approach presented in this paper is evaluated in the third DSTC, which focussed on the problem of state tracking in an extended domain, and is outperforms other approaches in tracking the user's *goal*.

Models which can adapt and improve with minimal labelling once they are deployed and exposed to dialogs in new domains are of particular interest. This is possible in dialog, as a system may be able to make inferences over sequences of observations, such as learning from user's clarifications. As a step in this direction, experiments are presented in adapting the neural network parameters online with no labels.

Several successful approaches for dialog state tracking have been entered and evaluated in the first two DSTCs, including maximum entropy models [3], web-style ranking [4], neural networks [5], robust hand-crafted rules [6], and conditional random fields [7, 8, 9]. The RNN framework presented here and originally in [10] provides a natural model for discriminatively learning over dialog sequences. The model is able to deal with simple but high-dimensional representations of dialog turns, requiring very little feature engineering. In particular the RNN framework allows for word-based state tracking, omitting the need for any intermediate semantic representation, which could be a further barrier to dialog in expanding domains.

2. RECURRENT NEURAL NETWORKS FOR DIALOG STATE TRACKING

A key component of the state of a dialog as it is formulated in the Dialog State Tracking Challenge is the user's *goal*. This is a mapping of *slots* to *values* and represents what constraints the user has given in the conversation so far. This section presents how the goal is tracked throughout the course of a dialog, using the recurrent neural network (RNN) framework described in [10].

2.1. Feature Representation of the Dialog Sequence

The first stage of the RNN approach consists of converting the input information into the features used by the model. The input components of each dialog turn are the system action and the user input (from a speech recogniser and/or a spoken lan-

guage understanding system). System actions and any spoken language hypotheses typically use some kind of meaning representation, which will be called a *dialog act*.

In the dialog state tracking challenge the dialog acts consist of a list of sub acts- each with an *act-type* and an optional slot, value assignment. An example is `hello() | inform(area=centre)` (*Hi I'm looking for something central.*), where the act-types are `hello` and `inform`. The dialog act features are obtained as a vector by expanding into n-grams. This example generates the n-grams `hello`, `inform`, `area`, `center`, `inform+area`, `area+centre`, and `inform+area+centre`.

Hypotheses from the automatic speech recognition (ASR) are converted to n-gram features for $n = 1, 2, 3$, following [11]. The set of hypotheses from the speech recogniser for a turn is represented as one vector by summing the individual vectors for each hypothesis, weighted by their probabilities. The set of spoken language understanding hypotheses is represented in the same way.

2.2. Delexicalised Features

Concatenating the vectors for the machine action and the user input gives a vector $\mathbf{f}$ which represents the turn. The RNN takes this $\mathbf{f}$ as input at each turn, and outputs a sequence of distributions over values for each slot. In order to allow for generalization to values and slots which are not seen in the training data, the approach taken here is to factor the neural network structure so that it operates on *delexicalised* features.

Delexicalised features are n-gram features where references to a particular slot or value have been replaced with a generic symbol. The procedure for creating the delexicalised vectors $\mathbf{f}_s$ and $\mathbf{f}_v$ is illustrated in figure 1. For example the delexicalised vector representations of the utterance "Jamaican food" for s=food and v=jamaican are very similar to those of "The Girton area" for s=area and v=girton. Therefore using these features as input allows the model to transfer learning from one case to the other, and achieve applicability to new slots and values.

To start tracking a new slot, it is only necessary to provide text forms for the slot name and its possible values for delexicalisation. The RNN structure described in the following section, after being trained on slots for which we have data, can then be applied to the new slot.

2.3. Network Structure

One RNN is used to track each slot, outputting a distribution $\mathbf{p}$ over all possible values, including a probability that the slot has not been mentioned yet. The memory state $\mathbf{m}$ and output $\mathbf{p}$ are updated to $\mathbf{m}'$ and $\mathbf{p}'$ respectively according to the recurrent structure shown in figure 2.

For each value v a scalar g_v is given by:

$$g_v = \text{NNet}\left(\mathbf{f} \oplus \mathbf{f}_s \oplus \mathbf{f}_v \oplus \{p_v,\ p_N\} \oplus \mathbf{m}\right)$$

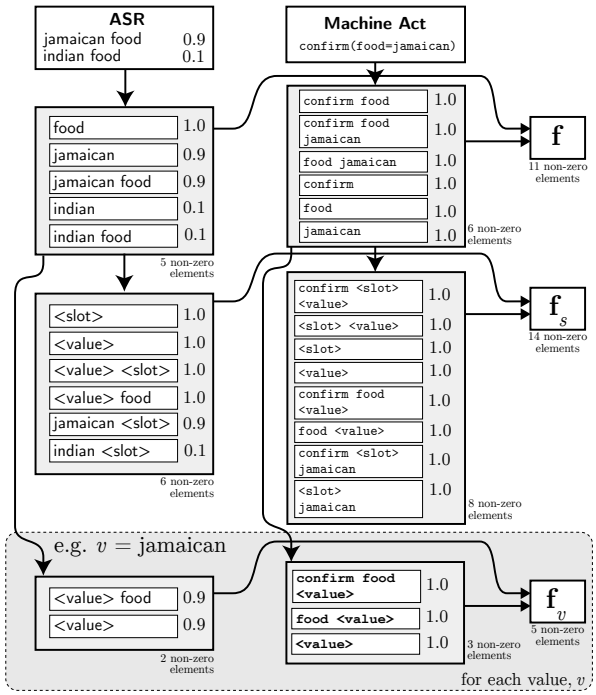

Fig. 1. Example of feature extraction for one turn, giving $\mathbf{f}$, $\mathbf{f}_s$ and $\mathbf{f}_v$. Here s=food. For all $v \notin$ {indian, jamaican}, $\mathbf{f}_v = \mathbf{0}$

where N is the number of values for the slot s, and $\oplus$ denotes vector concatenation. p_N is the probability assigned to the hypothesis that no value has been mentioned for s. The NNet() notation denotes a neural network with one hidden layer, which is then expanded in the final RNN structure. These scalars are then passed into a softmax to give the output distribution $\mathbf{p}'$:

$$\mathbf{p}' = \text{softmax}\left(\mathbf{g} \oplus \{B\}\right)$$

where B is a parameter of the RNN.

Here we do not include the component $\mathbf{h}$ described in [10] which adjusts $\mathbf{g}$ and is computed directly from $\mathbf{f} \oplus \mathbf{p} \oplus \mathbf{m}$. Since the parameters which $\mathbf{h}$ introduces are dependent on the number of possible values for the slot, N, omitting $\mathbf{h}$ reduces the dependence of the model on the ontology, and allows the model to be applied to any slot. In fact, as no particular assumption is made about the number of the possible values for the slot, the set of possible values can be dynamic by assuming $p_v = 0$ for values v with $\mathbf{f}_v = 0$.

Finally the memory $\mathbf{m}$ evolves in a standard manner, $\mathbf{m}' = \text{NNet}\left(\mathbf{f} \oplus \mathbf{m}\right)$

For training, the entire RNN is unravelled across all turns in a dialog sequence, expanding the NNet() components. Back-propagation is then used to minimise the log-probability of the training sequences in stochastic gradient descent.

3. ONLINE UNSUPERVISED ADAPTATION

This section presents an approach to adapt the RNN parameters in a live dialog system, without explicit labels, when

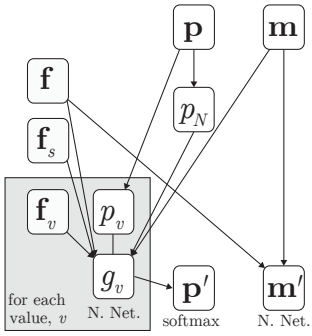

Fig. 2. Calculation of $\mathbf{p}'$ and $\mathbf{m}'$ for one turn and slot s.

deployed in a new domain. This is applied to trackers which work directly on the words from the speech recognition, with the hope that they might adapt to the language used to express constraints on the slots in the new domain.

Relevant fields in machine learning include semi-supervised learning [12], which attempts to learn from unlabelled in-domain data, and transfer learning [13], which tries to exploit labelled out-of-domain examples. The techniques presented in those fields are typically designed for static classification tasks. The approach defined here however exploits specific patterns of dialog sequences and is formulated specifically for use in live dialog state tracking.

In this context, models are initialised with a set of parameters W^{init}, and adaptation is the process of updating the parameters $W^\star$ having observed dialogs in the new domain. In online adaptation, $W^\star$ may be updated after each dialog, but the parameters may not be updated using any data from any dialog before the RNN has output its distributions for that dialog.

Let $\mathbf{f}_t$ denote the input features to the RNN at turn t, and define $\mathbf{F}$ as the sequence of inputs for a dialog: $\mathbf{F} = (\mathbf{f}_0, \ldots, \mathbf{f}_{T-1})$ where T is the length of the dialog. An RNN of a given topology can be considered as a function which takes a set of values for the weight and bias parameters, W, and maps $\mathbf{F}$ to a sequence of distributions over the labels, $\mathbf{Y} = (\mathbf{y}_0, \ldots, \mathbf{y}_{T-1})$. In particular, for the set of initial parameters W^{init},

$$\mathbf{Y}^{\text{init}} \quad = \quad \text{RNN}\left(W^{\text{init}}, \mathbf{F}\right)$$

Unsupervised adaptation is facilitated by defining a scoring criterion which evaluates the adapted parameters $W^\star$ without requiring any labels.

3.1. Criterion for Unsupervised Adaptation

The proposed criterion $C\left(W^\star\right)$ used to score a set of parameters $W^\star$, without requiring labels is defined as follows:

$$C\left(W^\star\right) = \left(\sum_{t=0}^{T-1} H(\mathbf{y}_t^{\text{init}}) H(\mathbf{y}_t^\star,\, \mathbf{y}_{t+1}^{\text{init}})\right) + \lambda \|W^\star - W^{\text{init}}\| \quad (1)$$

where $H(\mathbf{y})$ is the entropy of the distribution $\mathbf{y}$ and $H(\mathbf{y}, \mathbf{y}') = -\sum_i y_i \log(y_i')$ is the cross-entropy between $\mathbf{y}$ and $\mathbf{y}'$. $\|W\|$ is the norm of the the parameters W, e.g. the $l2$ or $l1$ norm.

The regularisation term multiplying $\lambda > 0$ enforces that $W^\star$ should stay close to W^{init}. In the sum, the only terms that change with $W^\star$ are the cross-entropies $H(\mathbf{y}_t^\star,\, \mathbf{y}_{t+1}^{\text{init}})$. As $\mathbf{y}^{\text{init}}$ is fixed, the cost is minimised if $\mathbf{y}_{t+1}^{\text{init}} = \mathbf{y}_t^\star$ at each t. Therefore in minimising C the parameters $W^\star$ receive a learning signal to adapt $\mathbf{y}_t^\star$ towards $\mathbf{y}_{t+1}^{\text{init}}$ which is weighted by $H(\mathbf{y}_t^{\text{init}})$.

When minimising C, at turns where the initial model is uncertain (high entropy), learning will attempt to find parameters that can predict the next turn's label according to the output of the initial model.

Figure 3 demonstrates an example dialog sequence where this leads to improved parameters, $W^\star$. In general the minimisation of C works to propagate a learning signal from later turns in the dialog, where an initial model should be more sure about the user's goal, to earlier turns. The term $H(\mathbf{y}_{t+1}^{\text{init}})$ ensures that this signal should stop at turns where the initial model is confident about its prediction.

Dialog Turn	$\mathbf{y}^{\text{init}}$	**Notes**
System: What type of food would you like? *User*: Chinese food.		Here an initial model is likely to output a confident low entropy distribution correctly identifying the food goal as 'Chinese'.
System: There are no matching Chinese restaurants. *User*: Any serving pizza?		The system has requested the food slot, and the user's response included the term 'serving'. This gives evidence that the user has informed the food slot, but the system cannot recognise which food type is correct. Therefore it is likely that an initial model would output a high entropy distribution for the food slot.
System: Sorry, what type of food would you like? *User*: Um, Italian food.		If the user explicitly says 'Italian', which the system is able to match in its ontology, then an initial model can predict with high confidence the correct value is food.

Fig. 3. Example dialog where user guidance can be exploited in unsupervised learning. The entropy of the food slot is high in the second turn, and low in the third peaking at 'Italian'. Therefore minimising C (equation 1) leads to a learning signal which correlates the n-grams in the second turn ('Something serving pizza') with the 'Italian' hypothesis for food. The low entropy of the slot in the first turn causes the corresponding term in C to diminish, so the signal does not lead to a correlation with the n-grams in the first turn.

3.2. Schedule for Online Adaptation

The following procedure performs online adaptation of the parameters while classifying a set of dialogs. This is used for

	Cardinality		
Slot	**Train**	**Test**	
type*	1	3	
area	5	15	*Old*
food	91	28	*slots*
name	113	163	
pricerange	3	4	
near	—	52	
hastv	—	2	*New*
hasinternet	—	2	*slots*
children-allowed	—	2	

Table 1. Summary of the smaller training set domain and the extended domain used for testing in DSTC3. The 'near', 'hastv', 'hasinternet' and 'children-allowed' slots are not found in the training data. (*)The 'type' slot denotes the type of venue the user is interested in, and is only ever 'restaurant' in the training set. In testing, type can also be 'coffee shop' or 'pub'.

tracking dialogs in the DSTC, and could also be used in a live system. It works by collecting dialogs in batches of size N, and updating the parameters using these batches.

1. Set $W^\star = W^{\text{init}}$ and $D = \emptyset$.
2. Track the next dialog using parameters $W^\star$. Append the dialog (represented by the input feature sequence $\mathbf{F}$) to the batch, $D \rightarrow D \cup \{\mathbf{F}\}$. If $|D| = N$, then enter step 3, otherwise return to step 2.
3. Update $W^\star$ using stochastic gradient descent to minimise $C(W^\star)$ over the dialogs in D. Reset D to $\emptyset$, and return to step 2.

4. RESULTS IN THE THIRD DIALOG STATE TRACKING CHALLENGE

The third Dialog State Tracking Challenge (DSTC3) studied the ability of trackers to adapt to an extended domain [14]. Training data was available in a smaller domain concerning restaurant information, while the test data included coffee shops and pubs as well as restaurants. The set of possible values for each slot was different in the test set, which also included extra slots. A summary of the difference between the smaller training domain and extended test domain of DSTC3 is given in table 1. Full details of the data and evaluation are given in [14].

The unlabelled test set for the DSTC3 was released over a one week period, at the end of which 7 different research groups submitted their tracker output for evaluation. This section explains how delexicalised recurrent neural networks were trained for this blind evaluation, and the results are presented. Following the evaluation, the use of unsupervised adaptation was investigated and the results of this subsequent study are presented in section 5.

Two trackers were submitted to DSTC3, one using speech recognition features (ASR) and another using both speech

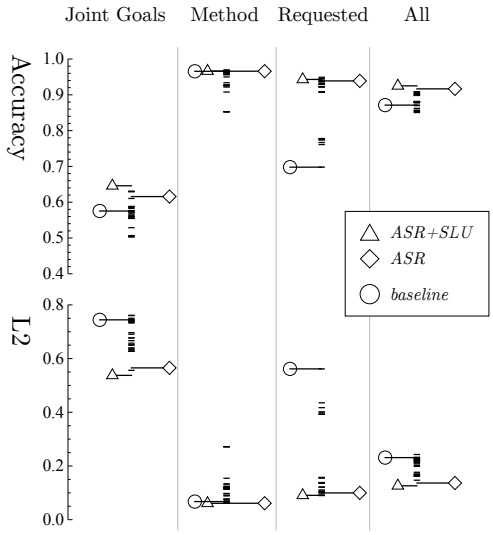

Fig. 4. Relative performance for featured metrics in DSTC3. Results from other teams are plotted as dashes. Note a lower L2 is better. The most competitive baseline system is shown as a circle.

recognition and spoken language understanding features (ASR+SLU). In both cases slot-independent RNNs were trained on the labelled data for all slots in the training set. The slot-independent models were used to track the new slots in the extended test domain. For slots which existed in the training set, slot-dependent models were trained by starting from the slot-independent model and continuing training using only the labelled data for that slot.

For each of the two trackers, an ensemble of six separate RNNs were trained with varying hyper-parameters[1], and then combined using score averaging.

Note that systems using only ASR features, also known as word-based trackers, are attractive as they do not require a spoken language understanding system for the new domain. A tracker which uses SLU features leaves open the question of how to design SLU systems for expanding domains, while word-based trackers avoid the need for this and require no intermediate semantic representations.

The challenge also required tracking *requested slots* and the *search method*, for which the approach in [10] was used. The performance in the DSTC3 evaluation is presented in table 2, and figure 4 plots the performance relative to the other teams' trackers. These results are identified under 'team3' in the DSTC3 overview paper [14].

The delexicalised RNNs performed consistently well across all tasks. The ASR+SLU tracker obtained the top accuracies and L2 scores for tracking the joint goal in the challenge. Among the trackers which did not use the SLU, the word-based ASR tracker obtained the top joint goal accuracy.

[1]The size of the memory $\mathbf{m}$ and the hidden layer in the neural network for g_v were varied for the ensemble.

	Joint Goals		Method		Requested	
	Acc.	L2	Acc.	L2	Acc.	L2
SLU+ASR	**0.646**	**0.538**	0.966	**0.061**	**0.943**	**0.091**
Top competing[1]	0.630	0.556	**0.970**	0.065	0.939	0.101
ASR	**0.616**	0.565	0.966	**0.061**	0.939	0.100
Top competing[2]	0.610	**0.556**	**0.968**	0.063	**0.949**	**0.090**

Table 2. Featured metrics in the third Dialog State Tracking Challenge (DSTC3). The L2 score is the square of the L2 distance between the tracker distribution and the true label (delta distribution), and so a lower score is better. The SLU+ASR and ASR trackers are identified as team3entry0 and team3entry2 respectively in the DSTC3 blind evaluation. The two rows labelled *Top competing* give the best results from the 6 other teams in the challenge (23 total entries) among trackers which [1]did and [2]did not use SLU features.

5. EVALUATION OF UNSUPERVISED ADAPTATION

This section evaluates the proposed method for online and unsupervised adaptation to extended domains in dialog state tracking. The method is initially validated using data from the second Dialog State Tracking Challenge (DSTC2) [2], emulating an extended domain by removing the labels for the 'food' slot during training. Results on testing in an extended domain are then presented on the DSTC3 data.

5.1. Adaptation with Missing Labels

Here adaptation is evaluated using data from a single domain. A set of initial trackers is trained without the labels for slot s, which are then adapted to the task of tracking the slot s. Though this is a fairly artificial setup, it may give us some confidence that adaptation will be beneficial during actual deployment in an extended domain (see next section). It is also conceivable that only partially labelled data may be available in some cases during training.

This study investigates training RNNs for the 'food' slot using only labels for 'area' and 'pricerange' in the DSTC2 training sets. Though the RNNs are never exposed to labels for food, they are used to classify the food goal in the DSTC2 test set. The food slot is chosen as it is an outlier in the ontology; while area and pricerange have cardinalities 5 and 3 respectively, there are 91 possible food types. This contributes to the food slot being the hardest to track – in the DSTC2 evaluation, trackers consistently scored lowest on food of all slots.

An ensemble of six RNNs, with varying hyper-parameters as described in section 4, were trained using ASR features and the labels for area and pricerange. Each of these can immediately be used to track the food slot without any unsupervised adaptation, which is called the *Unadapted* condition. Each of the six RNNs can also be adapted online during the classification of the test set using the procedure described in section 3, which is called the *Adapted* condition.

Figure 5 illustrates the performance of the six trackers with and without online unsupervised adaptation at test time. In all cases, adaptation improved accuracy on the test set. On average the accuracy improved by 1.4% and in the best case by 3.0%. The L2 score, which measures the quality of the probability scores was on average improved but in certain individual cases the L2 score deteriorated slightly.

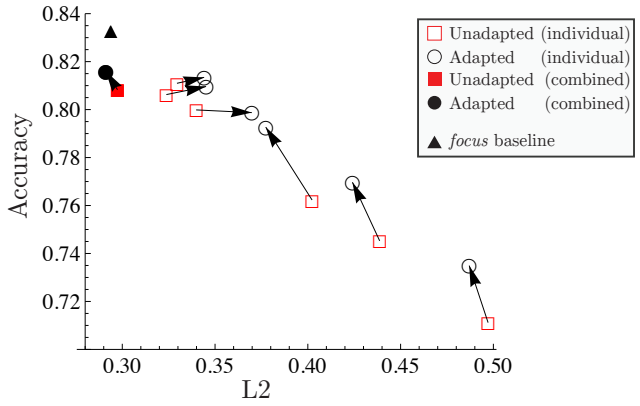

Fig. 5. Accuracy and L2 scores on the food slot for trackers trained using only labels for area and pricerange. Note lower L2 scores are better. The results for individual RNNs are shown as outlined shapes. Squares show the performance for initial RNN trackers (*Unadapted*) connected to circles showing the performance using unsupervised adaptation (*Adapted*). The performance for combining the groups using score averaging are also shown. The *focus* baseline was the strongest baseline in DSTC2. This baseline assumes a semantic decoder for the food slot, which the other trackers are not given.

The two groups of six were combined using score averaging. The combined results also demonstrate a slight improvement using adaptation, with the combined *Adapted* group giving the best tracking performance. The performance is comparable with many of the entries in DSTC2 including the *focus* baseline (the strongest baseline), and is achieved without any training labels for the slot.

5.2. Adaptation to New Domains

This section shows results in applying unsupervised adaptation to the word-based ASR tracker described in section 4.

Table 3 summarises the results on the DSTC3 test set. Unsupervised adaptation is shown to give an improvement in the mean accuracy for the old slots (those in the training data)

	New slots	Old slots	Joint
mean			
Unadapted	0.866	0.877	0.552
Adapted	0.869	**0.890**	**0.566**
combined			
Unadapted	0.893	0.900	0.616
Adapted	0.892	0.900	**0.623**

Table 3. Goal tracking accuracies on the DSTC3 test set for word-based RNN trackers, with and without unsupervised adaptation. Accuracy is reported on the *Old slots* and *New slots*, i.e. slots found and not found in the training set respectively. The joint goal accuracy is also given. In each condition, a group of six RNNs are trained. Mean accuracies for the group are reported, as well as the accuracies of the groups combined using model averaging. For *mean* results, bold denotes a difference of over 2 standard errors.

and the joint. After combination using score averaging, the top joint goal accuracy of 0.623 among the word-based ASR trackers is obtained by the combined adapted tracker. The resulting L2 score however is 0.587 – comparing this to table 2 it seems there is some trade-off between accuracy and the quality of the scores as probabilities, as measured by the L2 score, when using the unsupervised adaptation.

6. CONCLUSIONS

The RNN framework, when combined with delexicalised feature representations, provides a robust method for dialog state tracking which is able to generalise to unseen slots and values. The RNN tracker built using this approach was found to be one of the most competitive submitted to DSTC3, in particular, it outperformed all of the other systems in the task of tracking the user's goal.

The results of the study performed after the evaluation using unsupervised adaptation suggest that it is possible to exploit unlabelled dialog data to further improve RNN-based models online during deployment.

The methods presented here, when coupled with recent results in policy adaptation [15], suggest a technique for deploying dialog systems in expanding domains. Future work will evaluate these techniques in a live system, rather than in corpus tasks such as the DSTCs.

Acknowledgements

Matthew Henderson is a Google Research doctoral fellow.

7. REFERENCES

[1] Jason Williams, Antoine Raux, Deepak Ramachadran, and Alan Black, "The Dialog State Tracking Challenge," in *Proceedings of SIGDIAL*, 2013.

[2] Matthew Henderson, Blaise Thomson, and Jason Williams, "The Second Dialog State Tracking Challenge," in *Proceedings of SIGdial*, 2014.

[3] Jason Williams, "Multi-domain learning and generalization in dialog state tracking," in *Proceedings of SIGDIAL*, August 2013.

[4] Jason Williams, "Web-style ranking and slu combination for dialog state tracking," in *Proceedings of SIGDIAL*, 2014.

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

[6] Zhuoran Wang and Oliver Lemon, "A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information," in *Proceedings of SIGDIAL 2013*, 2013.

[7] Sungjin Lee and Maxine Eskenazi, "Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description," in *Proceedings of the SIGDIAL 2013 Conference*, 2013.

[8] Sungjin Lee, "Structured discriminative model for dialog state tracking," in *Proceedings of SIGDIAL 2013*, 2013.

[9] Hang Ren, Weiqun Xu, Yan Zhang, and Yonghong Yan, "Dialog state tracking using conditional random fields," in *Proceedings of SIGDIAL*, 2013.

[10] Matthew Henderson, Blaise Thomson, and Steve Young, "Word-based Dialog State Tracking with Recurrent Neural Networks," in *Proceedings of SIGdial*, 2014.

[11] Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young, "Discriminative Spoken Language Understanding Using Word Confusion Networks," in *Spoken Language Technology Workshop, 2012. IEEE*, 2012.

[12] Xiaojin Zhu and Andrew B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, 2009.

[13] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[14] Matthew Henderson, Blaise Thomson, and Jason Williams, "The Third Dialog State Tracking Challenge," in *Spoken Language Technology Workshop, 2014. IEEE*, 2014.

[15] Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young, "POMDP-based dialogue manager adaptation to extended domains," in *Proceedings of SIGDIAL*, 2013.