MULTI-DOMAIN DIALOGUE SUCCESS CLASSIFIERS FOR POLICY TRAINING

David Vandyke, Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Tsung-Hsien Wen and Steve Young

Department of Engineering, University of Cambridge, Cambridge, UK
{djv27, phs26, mg436, nm480, thw28, sjy}@cam.ac.uk

ABSTRACT

We propose a method for constructing dialogue success classifiers that are capable of making accurate predictions in domains unseen during training. Pooling and adaptation are also investigated for constructing multi-domain models when data is available in the new domain. This is achieved by reformulating the features input to the recurrent neural network models introduced in [1]. Importantly, on our task of main interest, this enables policy training in a new domain without the dialogue success classifier (which forms the reinforcement learning reward function) ever having seen data from that domain before. This occurs whilst incurring only a small reduction in performance relative to developing and using an in-domain dialogue success classifier. Finally, given the motivation with these dialogue success classifiers is to enable policy training with real users, we demonstrate that these initial policy training results obtained with a simulated user carry over to learning from paid human users.

Index Terms— statistical spoken dialogue systems, dialogue success, multi-domain, policy training

1. INTRODUCTION

By largely removing requirements to hand-craft components and rules, statistical approaches can significantly reduce development time in building spoken dialogue systems (SDS). In addition, learning to understand and respond from data has been demonstrated to produce more robust systems [2, 3].

However, in absolute terms, developing statistical SDS is still an expensive task. Even with the architecture in place, the requirements to obtain data for developing a dialogue system in a new domain, particularly if it requires supervised (expert) labels, may be a sticking point [4]. Indeed this cost may often be linear in the number of components of the SDS; e.g. different data may be required to individually train a belief tracker, dialogue manager and natural language generator. In order to move towards truly rapid development of SDS, methods and algorithms need to be discovered for making as much use of existing abilities of functioning SDS from other domains.

D. Vandyke and T-H. Wen are supported by Toshiba Research Europe Ltd, Cambridge Research Lab. P-H. Su is supported by Cambridge Trust and the Ministry of Education, Taiwan.

In assuming the framework of statistical SDS, we model dialogue as a sequential decision problem with a partially observable Markov decision processes [2, 5, 6] that explicitly accounts for inherent uncertainty in speech recognition and semantic understanding. In this context the dialogue manager (also 'policy' or 'agent') controls how the system responds and is trained by reinforcement learning (RL) [7], obtaining feedback from the environment via a reward function in order to evaluate and improve its behaviour.

Generally such agents are trained with a simulated user, or paid users presented with a task to complete. In such cases prior knowledge of the task is available in order to evaluate the agent's behaviour at the conclusion of each dialogue. Our goal however is to be able to learn from real users, by which we mean humans who come to a SDS with their own goals in mind, and in this case it is a difficult task to specify the reward function as the users goal is unobserved. As such inference on the users goal or directly on the dialogue's success is required.

Of course in some situations it is possible and even natural for the system to ask the user for feedback at the dialogues conclusion (e.g. confirming a purchase with a *yes/no* question), however this is not always possible and user responses can be noisy [8] which results in slower learning.

In [1] a supervised neural network (NN) model was proposed for inferring whether a dialogue was successful (*did the user obtain what they were after from the system?*) or not based only on information available in the 'real user' learning environment. That is it inferred the dialogue's binary success/failure label without requiring unavailable knowledge of what goal the user had in mind, doing this with $\sim 95\%$ accuracy. Policies trained with this model as a reward function were subsequently adjudged by human's evaluations to be as good as one trained when using the prior task knowledge.

The model as described in [1] however was limited to being able to operate only within the SDS domain it was trained in. Because the model is non-trivial to develop, and because we would like to develop new SDS with minimal effort, we would ideally like to be able to use the developed model to classify dialogues in new, unseen domains. This is the focus of this paper. Our main contributions are to:
a) Reformulate the feature inputs to the model such that it has the possibility of operating across multiple domains.
b) Show evidence that when no labelled data is available in a

new domain, that models from other domains are able to directly make accurate predictions.
c) Show that when data is available in a new domain generally more accurate models can be built by pooling training datasets rather than building a model from scratch or performing model adaptation.
d) Importantly demonstrate that these multi-domain models are able to train comparatively good policies in new domains.

The paper is organised as follows: §2 reviews the RNN model proposed in [1] and an overview of the SDS we experiment with is given in §3. In §4 we outline our proposal for overcoming the difficulties in directly using or adapting the model in [1] to new domains. The four domains used in our experiments are also described along with results of predicting dialogue success/failure outcomes. How these models subsequently perform when used to inform reward functions for policy training is then investigated in §5. Related research is discussed in §6 and conclusions are drawn in §7.

2. REVIEW: RNN DIALOGUE SUCCESS MODEL

We now review the Recurrent NN (RNN) model introduced in [1]. This is a vanilla RNN that takes as input a feature extracted at each turn (system + user exchange) of the dialogue, and updates a hidden layer forming a representation of the observed sequence to date. This continues until the dialogue's completion at which time it outputs a probability distribution over the binary target labels conditioned on its hidden representation. This is a standard Elman RNN topology with which we use a cross-entropy loss function and sigmoid activations in all neurons and that is trained with a dataset of input-output pairs via backpropagation, see e.g. [9] for details.

We note also that several other recurrent architectures were investigated, but none found to be consistently more accurate than the vanilla RNN model we use. These included LSTM [10], GRU [11] as well as bi-directional [12] and deep/stacked [13] versions of each of these (results omitted due to space). Our hypothesis is that for this task, the most informative features occur towards the dialogue's conclusion, such that the model can afford to 'forget' earlier inputs [14]. Some evidence for this is presented in Table 1, where only a small reduction in the model's F-measure are observed when making predictions based on only the final 5 turns rather than all turns in the dialogue.

	`testA`	`testB`
Whole Dialogue	94.1	90.4
Last 5 Turns Only	94.0	90.0

Table 1: Comparison of F-measures of RNN dialogue success prediction models when using *all* versus *only the final five turns*. **TT** domain. Models trained on `train-1K`. The domains and train/test datasets are described in §4.2.

3. DIALOGUE SYSTEM DESCRIPTION

In all experiments the following SDS components were used. Belief updating was done via the BUDS [15] dynamic Bayesian network operating over state and user goal variables. Policy learning was implemented by a Gaussian process via the GP-SARSA algorithm [16]. As policies typically require $\mathcal{O}(10^4)$ dialogues to converge, a simulated user [17] (operating at the semantic level) was used prior to the concluding human user trial. The natural language understanding and generation components used for human user trials were both hand crafted, using a Phoenix grammar [18] and templates (mapping system semantics to natural language) respectively.

Policies were trained with the following reward function:

$$\mathcal{R}(\mathcal{D}) = 20 * \mathbb{1}(\mathcal{D}) - N \quad (1)$$

where N is the number of turns in dialogue $\mathcal{D}$, and $\mathbb{1}(\mathcal{D})$ is an indicator function for the dialogues success. $\mathcal{R}(\mathcal{D})$ is given at the dialogue's end and a reward of 0 is given at all preceding turns. $\mathbb{1}(\mathcal{D})$ is determined by either an RNN model's prediction or, for the baselines, given by an objective measure that uses exact prior knowledge of the user's goal.

4. MULTI-DOMAIN SUCCESS PREDICTION

As mentioned, the model described in [1] is somewhat expensive to develop, requiring the development of a supervised learning dataset of input-output pairs and the subsequent training. Given a new domain in which we wish to develop a SDS, we would like to be able to directly use (or at least bootstrap) our prior efforts in developing earlier dialogue success classifiers in order to rapidly obtain suitable classification performance in this new domain.

The central issue in [1] that precludes this is that the turn-level features fed to the RNN to make a success/failure classification are of domain-dependent dimensions. We are thus outside of the standard machine learning adaptation problem, which may be posed as: given a distribution over inputs $P(x)$ and a classifier (conditional distribution over targets) $P(y|x)$, how do we adjust our classifier when the input distribution changes?

The core issue for developing multi-domain RNN success classifier models is thus dealing with this input feature variability and establishing a common input feature space.

4.1. Features: multi-domain representations

In order to obtain performance across multiple domains with a single model we look to develop fixed length features via making simple approximations to the previously introduced [1] domain dependent feature, here denoted by $\mathcal{F}$.

To review, $\mathcal{F}$ is extracted at each turn by concatenating the following: one-hot encodings of each of the system action and

most likely user action (as estimated by the belief tracker) $\oplus$ full belief state (distributions over slot values) $\oplus$ turn number normalised by the maximum number of turns (here 30).

The dimension of $\mathcal{F}$ thus contains two independent sources of domain dependent variability: the number of slots and slot cardinality (the number of values within a slot). We propose to deal with the cardinality by replacing the full distributions over each slot ('slot beliefs'), by some predefined set of summary statistics of each distribution. Here we use simply the normalised entropy.

The remaining variability pertains to the differing number of slots across domains. For the system we experiment with, there is a fixed set of N_a system actions available for each slot $\mathcal{S}$ (e.g. '*request-*$\mathcal{S}$', '*confirm-*$\mathcal{S}$', etc.). If we have N_s slots, then our variability is in the fact that we have N_s slot entropies, plus $N_s \times N_a$ slot dependent system actions.

This is dealt with in two ways. The first is to make a coarse approximation whereby the slot distribution information is discarded and all slot dependent system actions are mapped to a single slot-independent action (e.g. '*request-*$\mathcal{S}_1$', ..., '*request-*$\mathcal{S}_{N_s}$' $\mapsto$ '*request*'). This fully domain-independent feature is of length 28 in our experiments, and we denote it by $\mathcal{F}_{28}$. Diagrams are given in the appendix §9.

The second approach is to impose a limit on the number of slots a domain can have, and pad out (with zeros) the slot distribution summary (normalised entropy) components along with the domain dependent system actions up to this limit. We set a limit in our experiments of a maximum of 6 (user) informable slots, making this feature of length 74, so we denote it by $\mathcal{F}_{74}$. It is important to maintain an order to the components of this representation still, such that the RNN models have some chance to generalise to new domains.

Note that the raw features $\mathcal{F}$ certainly do contain much redundancy. Using an autoencoder (AE) NN to perform dimension reduction resulted in almost no loss in the RNN model's performance (not reported). However, whilst one can map features from multiple domains into a common space with such methods, there is no expectation for obtaining similarly distributed inputs across domains. This proved to be the case. Dimension reduction methods only reformulate the problem as a standard machine learning adaptation problem.

4.2. Domains and datasets

Four individual domains are used to explore questions regarding the performance of the RNN models. These are slot-value type ontologies describing: 1) restaurant information within Cambridge, UK called Toptable (**TT**), 2) San Francisco Restaurants (**SFR**) and 3) SF Hotels (**SFH**), and lastly 4) a laptop information system (**LAP**).

Table 2 outlines the size of the domains, as well as the size of the raw feature $\mathcal{F}$ given to the RNN model at each turn per [1]. The size of this feature implicitly gives an indication of the cardinality ranges of the slots; both **SFR** and **LAP** have 6 '*informable*' (from the users perspective) slots, but the raw **SFR** feature is much larger as **SFR** contains slots with a greater number of possible values.

Domain Key:	**TT**	**SFR**	**SFH**	**LAP**
# Informable Slots:	3	6	6	6
# Requestable Slots:	3	3	3	3
Raw RNN Feature $\mathcal{F}$ Size:	617	1167	970	497

Table 2: Domain identifiers, sizes and size of domain dependent raw RNN classifier feature $\mathcal{F}$. See text for details.

All datasets for training RNN dialogue success models are obtained from training policies from random with a user simulator [17], producing supervised pairs: (sequence of turn level dialogue features, objective success/failure target label). The semantic error rate (SER) of the simulated user is set to 15% and data is balanced regarding target labels.

For each domain we produced a training set of 18K dialogues (`train-18K`) and a smaller set of only 1K (`train-1K`). A separate validation set of 1K dialogues is used with both training sets to control overfitting. Two test sets were used: a further 1K dialogues also at SER 15% (`testA`) and a larger set (`testB`) containing 3K dialogues from each of four SER (0%,15%,30%,45%) as the data occurred (i.e. by training policies from random in each SER condition and without balancing success/failure targets).

The `train-1K` condition is more realistic than `train-18K` in the sense that dialogues may be collected in some other way and require success/failure annotations by human experts. The training data may also only be collected in a single environment, so `testB` gives an indication of how models may perform in real world applications which can experience a variety of environmental factors that influence semantic understanding or speech recognition rates.

4.3. Single domain models

Before investigating basic generalisation properties of the RNN success/failure classifiers[1], we report in-domain only results using the original feature $\mathcal{F}$ as well as the two fixed length features $\mathcal{F}_{28}$ and $\mathcal{F}_{74}$. These are given as F-measures (which are more informative than accuracy) in Table 3, with all results reported as the average of three models (each RNN beginning from a different random initialisation of their weights) along with one standard error. The binary success/failure label is taken as the most probable given the distribution over labels output by the RNN.

Inspection of these results using $\mathcal{F}$ in each domain provides evidence that the method proposed in [1] is viable, although we note that the highest performance is consistently

[1] All models were built in Python using Theano [19, 20]. Stochastic gradient descent per dialogue was used during backpropagation to train each model, and almost no optimisation of hyper-parameters (learning rate, model structure) was performed. All hidden layers where set to half the input feature size. A cross-entropy loss function was used with sigmoid activations everywhere. L1 and L2 regularisation were found unnecessary.

Domain:	TT			SFR		
Features:	$\mathcal{F}$	$\mathcal{F}_{28}$	$\mathcal{F}_{74}$	$\mathcal{F}$	$\mathcal{F}_{28}$	$\mathcal{F}_{74}$
`train-18K + testA`	$96.0 \pm .0$	$95.8 \pm .1$	$95.7 \pm .2$	$92.3 \pm .4$	$89.8 \pm .8$	$92.4 \pm .6$
`train-1K + testA`	$95.0 \pm .0$	$93.6 \pm .4$	$94.1 \pm .2$	$90.8 \pm .1$	$90.0 \pm .0$	$89.6 \pm .0$
`train-18K + testB`	$94.0 \pm .0$	$92.8 \pm .1$	$92.9 \pm .2$	$88.0 \pm .8$	85.5 ± 2.4	$89.6 \pm .5$
`train-1K + testB`	$91.0 \pm .0$	$90.5 \pm .3$	$90.4 \pm .2$	$89.2 \pm .1$	$86.0 \pm .0$	$87.1 \pm .1$
Domain:	**SFH**			**LAP**		
Features:	$\mathcal{F}$	$\mathcal{F}_{28}$	$\mathcal{F}_{74}$	$\mathcal{F}$	$\mathcal{F}_{28}$	$\mathcal{F}_{74}$
`train-18K + testA`	$94.3 \pm .1$	$92.2 \pm .5$	$93.8 \pm .1$	$91.8 \pm .3$	$85.5 \pm .3$	$90.9 \pm .5$
`train-1K + testA`	$90.7 \pm .1$	$89.3 \pm .1$	$89.8 \pm .0$	$84.8 \pm .1$	$83.4 \pm .1$	$88.0 \pm .3$
`train-18K + testB`	$91.4 \pm .1$	$90.6 \pm .7$	$91.2 \pm .2$	$85.1 \pm .3$	$82.1 \pm .5$	$81.4 \pm .5$
`train-1K + testB`	$88.9 \pm .1$	$87.4 \pm .3$	$86.6 \pm .2$	$81.7 \pm .0$	$77.8 \pm .1$	$77.5 \pm .7$

Table 3: Shown for all 4 domains are model F-measures on the 2 in-domain test sets (`testA`, `testB`) when using each of the 3 considered turn level inputs: $\mathcal{F}, \mathcal{F}_{28}$ and $\mathcal{F}_{74}$. Models are trained on either `train-1K` or the larger `train-18K`.

achieved within the smallest domain, namely **TT**. The results also demonstrate only a small but persistent decrease in performance between training on `train-18K` and `train-1K`.

Comparing columns within each domain we note that models using the full feature $\mathcal{F}$ are most accurate, but that the use of the coarser, fixed length inputs typically resulted in only a small reduction in the F1 measure. This small degradation is the cost we incur when enabling multi-domain functionality via these feature adjustments.

For all subsequent experiments we stick to training models with the $\mathcal{F}_{74}$ feature as it is typically the most accurate of the two features allowing multi-domain capabilities. We also subsequently only consider training with the `train-1K` datasets, as requiring data of this volume is more realistic for developing models in situations where human labelling of gathered dialogues may be required in order to construct the training sets, and should therefore broaden their appeal.

4.4. Multi-domain models

4.4.1. Case 1: Generalisation

When we have no data in a new domain for training an RNN success predictor, we would ideally like to be able to directly use an existing model from another domain. The *upper* portion of Table 4 shows the performance of models (rows) trained with `train-1K` using $\mathcal{F}_{74}$ and tested both within their own domain[2], and directly in each of the other three domains. Again three models are trained in each domain and results are averages across the three on each test set. Standard errors are omitted due to space, but were of a similar negligible order as those reported in Table 3. We note that the models perform remarkably well in completely new domains, as emphasised by the final column where the arithmetic average of the F-measure of each model across all four domains is given. Recall that the chance rate is 50% on the balanced `testA` dataset, while approximately 70% of targets are 'success' labels in the unbalanced `testB` set.

4.4.2. Case 2: Pooling & Adaptation

If a dataset D with labelled targets exists in a new domain, the question then becomes how best to use this data? We have at least three **options**: **(1)** we can use the data to train a model, from a random initialisation (RI) of the RNN weights, as normal; **(2)** if data is available from previous model development in another domain we can combine or pool this with D and train a multi-domain model (again from RI); **(3)** we can use D to adapt a model from a different domain, meaning perform stochastic gradient descent (training) with D using the previous model as a starting point (rather than from RI).

First we consider options **(1)** and **(2)**. **(1)** may be considered a baseline and is given by the diagonal entries of the *upper* part of Table 4. Results of pooling data, option **(2)**, are given in the *lower* part of Table 4. Pooling choices, given space, do not cover all possible combinations but are done based on semantic similarities of the domains (restaurants, or hotel + restaurants), except for the final row in which case all four domains `train-1K` data are simply pooled together.

Regarding option **(3)** of adaptation, Table 5 shows the results in each individual domain when taking a model from another domain as an initialisation of RNN weights and performing backpropagation with the new domains data. All results in this table are within the same domain as training and are averages of adapting from three in-domain models each trained from a different random initialisation. Note that the diagonals are purely option **(1)** RI models.

Comparing pooling and adaptation results we note that both approaches generally produce more accurate models compared to isolated, in-domain model building (option **(1)**). Analysing Tables 4 and 5 we see that pooling typically produces more accurate models than adaptation, although the differences are small. Given this result however, we don't pursue the use of any adaptation models for policy training.

[2]Note that the in domain results in Table 4, the 'diagonal' entries in the *upper* section, correspond exactly to the appropriate elements of Table 3.

	TT		SFR		SFH		LAP		AVERAGE	
	testA	testB	testA	testB	testA	testB	testA	testB	testA	testB
Single:										
TT	94.1	90.4	82.3	88.8	83.0	88.7	78.5	84.2	84.5	88.0
SFR	85.5	86.2	89.6	87.1	89.9	88.3	83.6	78.2	87.2	85.0
SFH	85.3	85.6	90.0	85.3	89.8	86.6	83.7	76.2	87.2	83.4
LAP	85.3	82.7	88.7	84.1	87.7	84.7	88.0	77.5	87.4	82.2
Pooling:										
TT + SFR	94.0	91.3	91.8	90.8	90.9	91.1	85.1	83.1	90.5	89.1
TT + SFR + SFH	93.9	91.7	89.5	87.6	91.8	88.8	86.0	80.0	90.3	87.0
All 4 domains	93.3	90.8	89.9	87.4	89.8	88.0	87.9	80.0	90.2	86.5

Table 4: **Generalisation** - *Upper section:* models trained on their `train-1K` sets and *Lower section:* trained on the pooling of these datasets. Columns show the F-measure of the resulting models for `testA` and `testB` in all four domains.

Adapted from ↓	TT	SFR	SFH	LAP	Avg.
TT-model	90.4	87.6	88.5	76.1	85.7
SFR-model	90.7	87.1	90.4	78.7	86.7
SFH-model	91.6	86.8	86.6	75.8	85.2
LAP-model	90.7	85.9	88.1	77.5	85.5

Table 5: **Adaptation** - Success label F-measure on `testB` for models trained with `train-1K` when adapted from a model already trained in another domain.

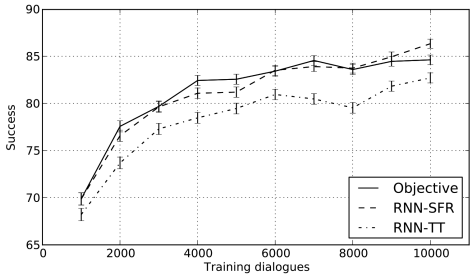
5. POLICY TRAINING

We now explore how the RNN model's performance in predicting success/failure labels translates to performance on the policy training task, where the binary label output by the RNN is used directly as the dialogue success indicator $\mathbb{1}$ in Eqn. 1.

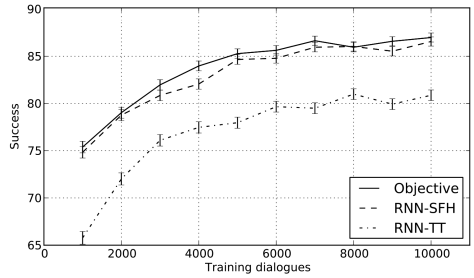
5.1. Policy training: simulate user

To gain an initial insight into the ability of these RNN dialogue success classifiers for policy training, in Figure 1 policy training results are shown in the **SFR** and **SFH** domains when training with a simulated user and comparing three different reward signals: a baseline given by the objective reward which makes use of exact knowledge of the users goal, and two RNN success predictors, one built in-domain in each case and the other directly using the success classifier model trained in the **TT** domain.

It can be seen that the use of the in-domain RNN model in each case is able to produce policies which are within one standard deviation of the baseline policy for each domain as trained under its objective reward signal. Regarding generalisation of the RNN models, it is seen that direct use of the out-of-domain **TT** RNN model is able to train comparable policies but introduces a loss of between 2% and 5% absolute against the baseline in both cases.[3]



(a) Simulated user policy training in: **SFR**



(b) Simulated user policy training in: **SFH**

Fig. 1: Policy training results in (a) **SFR** and (b) **SFH** when using either the objective reward (baseline), in-domain RNN classifier or directly using the out-of-domain **TT** RNN model.

With this initial insight into how the performance of the RNN models translate into policy training performance (as measured against the objective baseline) we now consider using the RNN models trained on data pooled across multiple domains, which were shown to produce the most accurate success classifiers. Figure 2 shows simulated user policy training results in the **LAP** domain with the objective baseline reward signal, compared to using an RNN model trained on the pooled data of the 3 other domains, and to the RNN model trained on the pooling of all four domains (the models in the 2nd last and last rows of Table 4 respectively). We subsequently refer to these models as `pool3` and `pool4`.

[3]In both Figure 1 and Figure 2 all policies are evaluated after every 1000 training dialogues all under the same objective measure with 1000 testing dialogues. Five policies are trained in all conditions and results are reported as averages over these. All training occurred at an SER of 15%.

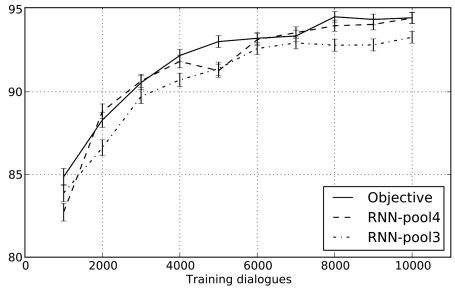


Fig. 2: **LAP** policy training under reward functions given by the objective (baseline), and `pool3` and `pool4` RNNs.

It is seen that greater success prediction accuracy translates to improved policies in the expected manner. Also by comparing Table 4 with the Figures 1 and 2 it is seen that RNN models which are only $\sim 90\%$ accurate at predicting success/failure labels for dialogues are still able to be used to train policies which suffer a reduction of only a few percent absolute as measured against the baseline policies trained under the objective reward function which is unavailable in the 'real user' learning scenario.

5.2. Policy training: human users

The `pool3` and `pool4` RNNs were then used to train a policy in **LAP** with human users recruited via Amazon Mechanical Turk. By doing so we investigated whether the simulated user policy training results could translate to learning from humans. No objective reward function was used as a baseline as it was demonstrated in [1] that in-domain RNNs were able to train policies with real users that are not statistically different to the ones trained under the objective baseline. Tasks required finding a laptop meeting 3 given constraints and subsequently querying a further property of the retrieved laptop.

We train 3 policies for both cases, collecting 300 dialogues in all 6 policies. To evaluate the resultant policies learning was stopped and subjective feedback was collected from the users regarding their ability to complete 50 further tasks with each of the 6 policies. Averages of the user feedback over the 150 tasks covered by each of the `pool3` and `pool4` systems are given in Table 6. Observing the subjective *Success*, we see the system trained with the `pool3` model that has not seen any **LAP** data is able to train a policy that is comparable to using the `pool4` model. The *Quality* of the dialogues were adjudged to be slightly inferior however.

	`pool3`	`pool4`
Quality (0-5)	3.16 ± 0.11	3.45 ± 0.11
Success (%)	72.2 ± 3.7	78.7 ± 3.3

Table 6: Human evaluations of **LAP** policies trained with the out-of-domain `pool3` RNN, and with the `pool4` model. *Quality*: 6-point Likert scale rating, *Success*: binary label.

6. RELATED WORK

In machine learning in general much research has looked at adaptation of statistical models [21, 22, 23] however research into adaptation of SDS components to new domains [24, 25, 26, 27, 28] or user behaviour [29] presents its own challenges and is comparatively nascent. Research into these questions is growing though [30], and will continue to given the natural progression towards multi-domain SDS [31, 32, 33].

A lot of work has looked at methods and metrics for evaluating SDS [34, 35, 36]. These have generally been considered as aids to system developers to experiment with design choices and recognise those that are leading to certain measures of good performance. Typically these require annotations or features which preclude their use directly in RL reward functions for direct training of dialogue managers in SDS. Certainly one advantage of RL training is that, if a reward function can be specified that encodes the problem well, then the policy can be both trained and evaluated under it.

Methods that, similar to our proposed approach, fit this paradigm are found in [37] where a corpus of dialogues with expert annotations of dialogue success are used for developing a turn-level reward function suitable for policy training, and in [4] where the environment external to the agent is developed from Wizard-of-Oz data (including a reward function) in which subsequent RL policy training is performed.

7. CONCLUSIONS

We have shown that the proposed turn-level inputs given to the RNN dialogue success classifiers enable strong multi-domain performance and generalisation abilities depending upon whether training data is or is not available within a new domain respectively. These ratings provided by the RNN models were also importantly shown to provide suitable feedback for RL training of SDS policies in both simulated and real user scenarios.

Although initially we considered exploring more sophisticated methods for handling the variable number of slots (e.g. recursive autoencoders [38], or mapping to a distribution over some certain discovered or defined slot archetypes), we quickly discovered that making approximations to the RNN inputs resulted in good models. Of the two proposed features enabling multi-domain capabilities, since Table 3 shows that the fully domain independent feature $\mathcal{F}_{28}$ is typically less accurate than $\mathcal{F}_{74}$, one remaining concern is with the imposed restriction of $\mathcal{F}_{74}$ on the number of allowable slots in a domain. Several options may be considered here but we predict that the most beneficial approach is to only consider the top N slots in a domain as ranked by the entropy of the domain's database of entities. We are currently developing larger domains in order to answer these and other questions, such as can active reward learning [39] be introduced into the proposed framework to improve reward function accuracy.

8. REFERENCES

[1] Pei-Hao Su, David Vandyke, Milica Gašić, Dongho Kim, Nikola Mrkšić, Tsung-Hsien Wen, and Steve Young, "Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems," in *Interspeech*, 2015.

[2] Jason D. Williams and Steve Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[3] Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin, "A comprehensive reinforcement learning framework for dialogue management optimization.," *J. Sel. Topics Signal Processing*, vol. 6, no. 8, pp. 891–902, 2012.

[4] Verena Rieser and Oliver Lemon, "Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets," *Comput. Linguist.*, vol. 37, no. 1, pp. 153–196, 2011.

[5] Nicholas Roy, Joelle Pineau, and Sebastian Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proceedings of ACL*, 2000.

[6] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 1, pp. 11–23, Jan 2000.

[7] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1999.

[8] L.B. Larsen, "Issues in the evaluation of spoken dialogue systems using objective and subjective measures," in *ASRU*, 2003, pp. 209–214.

[9] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville, "Deep learning," Book in preparation for MIT Press http://www.iro.umontreal.ca/ bengioy/dlbook, 2015.

[10] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[12] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, 1997.

[13] Alex Graves, "Generating Sequences With Recurrent Neural Networks," *CoRR*, vol. abs/1308.0850, 2013.

[14] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[15] B. Thomson and S. Young, "Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems.," *Computer Speech and Language*, vol. 24, pp. 562–588, 2010.

[16] Milica Gašić and Steve Young, "Gaussian processes for pomdp-based dialogue manager optimisation," *TASLP*, vol. 22, 2014.

[17] J. Schatzmann and S. Young, "The hidden agenda user simulation model," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 17, no. 4, pp. 733–747, May 2009.

[18] Wayne Ward, "Extracting information in spontaneous speech," in *ICSLP*. 1994, ISCA.

[19] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[20] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS Workshop, 2012.

[21] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh, "Domain adaptation: Learning bounds and algorithms," *CoRR*, vol. abs/0902.3430, 2009.

[22] Matthew E. Taylor and Peter Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, 2009.

[23] Hal Daume III, "Frustratingly easy domain adaptation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic, June 2007, pp. 256–263, Association for Computational Linguistics.

[24] K. Georgila and O. Lemon, "Adaptive multimodal dialogue management based on the information state update approach," *W3C Workshop on Multimodal Interaction*, vol. 23, pp. 142, 2004.

[25] Verena Rieser and Oliver Lemon, *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*, Springer, 2011.

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

[27] Milica Gašić, Dongho Kim, Pirros Tsiakoulis, and S. Young, "Distributed dialogue policies for multi-domain statistical dialogue management.," in *IEEE ICASSP, Brisbane.*, 2015.

[28] Nikola Mrkšić, D O Seaghdha, B Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young, "Multi-domain dialog state tracking using recurrent neural networks.," in *ACL*, 2015.

[29] Diane J. Litman and Shimei Pan, "Designing and evaluating an adaptive spoken dialogue system," *User Modeling and User-Adapted Interaction*, vol. 12, no. 2-3, pp. 111–137, 2002.

[30] O. Lemon and O. Pietquin, "Introduction to special issue on machine learning for adaptivity in spoken dialogue systems," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 7, no. 3, pp. 3, 2011.

[31] Guanchun Wang Hao Tian Hua Wu Zhuoran Wang, Hongliang Chen and Haifeng Wang, "Policy learning for domain selection in an extensible multi-domain spoken dialogue system," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[32] Donghyeon Lee, Minwoo Jeong, Kyungduk Kim, Seonghan Ryu, and Gary Geunbae Lee, "Unsupervised spoken language understanding for a multi-domain dialog system," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 21, no. 11, pp. 2451–2464, 2013.

[33] Mikio Nakano, Shun Sato, Kazunori Komatani, Kyoko Matsuyama, Kotaro Funakoshi, and Hiroshi G. Okuno, "A two-stage domain selection framework for extensible multi-domain spoken dialogue systems," in *Proceedings of sigDial*, 2011, pp. 18–29.

[34] P. J. Price, "Evaluation of Spoken Language Systems: The ATIS Domain," in *Speech and Natural Language Workshop*, 1990, pp. 91–95.

[35] Zhaojun Yang, Gina-Anne Levow, and Helen M. Meng, "Predicting user satisfaction in spoken dialog system evaluation with collaborative filtering," *J. Sel. Topics Signal Processing*, vol. 6, no. 8, pp. 971–981, 2012.

[36] Sebastian Möller, Klaus-Peter Engelbrecht, and Robert Schleicher, "Predicting the quality and usability of spoken dialogue services.," *Speech Communication*, vol. 50, no. 8-9, pp. 730–744, 2008.

[37] Layla El Asri, Romain Laroche, and Olivier Pietquin, "Reward Function Learning for Dialogue Management," in *Proceedings of the sixth Starting Artificial Intelligence Research Symposium (STAIRS)*, 2012, pp. 95 – 106.

[38] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *ICML*, 2011, pp. 129–136.

[39] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters, "Active reward learning," in *Robotics: Science and Systems X*, 2014.

9. APPENDIX

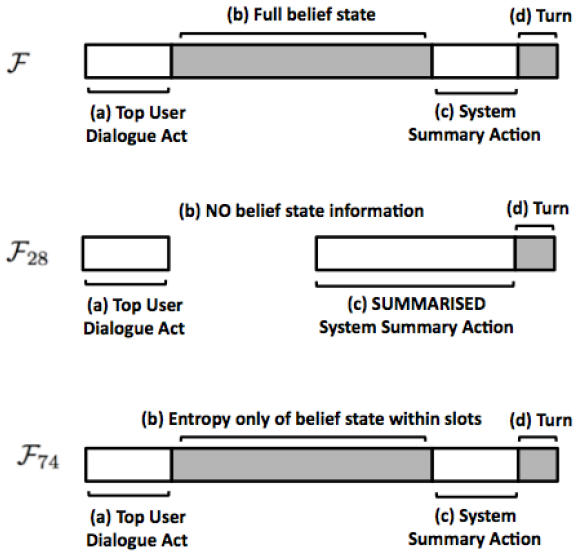


Fig. 3: **Feature diagrams**. $\mathcal{F}$: the original (domain dependent) feature as proposed in [1], $\mathcal{F}_{28}$: the coarse approximation which discards all belief state information and summarises slot-dependent system actions into a master system action (e.g.'*request-*$Slot_1$', ..., '*request-*$Slot_{N_s}$' $\mapsto$ '*request*'), and $\mathcal{F}_{74}$: which first imposes a limit on the number of (user) informable slots and then retains only an entropy statistic of each slot's belief distribution, as well as retaining all slot-dependent system summary actions (e.g.'*request-*$Slot_1$', ..., '*request-*$Slot_{N_s}$'), padding with zeros the segments labeled (b) and (c) if the current domain has less slots than the imposed limit on the number of slots, in order to maintain a fixed length vector - 74 dimensional within the reported experiments of this paper). See §4.1 for details.