Incremental on-line adaptation of POMDP-based dialogue managers to extended domains

M. Gašić, D. Kim, P. Tsiakoulis, C. Breslin, M. Henderson, M. Szummer, B. Thomson, S. Young

Cambridge University Engineering Department, Trumpington Street, Cambridge, UK

mg436@cam.ac.uk

Abstract

An important property of open domain spoken dialogue systems is their ability to deal with a set of new, previously unseen, concepts introduced in the conversation. The dialogue manager must then quickly learn how to talk about the new concepts using its knowledge of the existing concepts. It has previously been shown that a single new concept could be accommodated by mapping the kernel function of a Gaussian process to incorporate an additional concept into the domain of a statistical dialogue manager. Here we present an incremental scheme which enables the domain of a dialogue manager to be repeatedly extended by recursively specifying priors in Gaussian processes. We show that it is possible to effectively double the number of concepts understood by a system providing restaurant information using only 1000 adaptation dialogues with real users.

Index Terms: spoken dialogue systems, Gaussian processes

1. Introduction

In recent years there has been significant improvement in the area of limited domain goal-directed spoken dialogue systems. The introduction of statistical methods has been shown to provide more robust performance and be capable of learning from data thereby avoiding the need for hand-crafting dialogue decisions [1, 2, 3, 4, 5, 6]. In order to support large and potentially open domains, techniques are needed to automatically extend the operation of a dialogue system to cover previously unseen concepts such as a new slot name and the values which can fill that slot. This adaptation must be robust to mismatched training data [7] and support adequate user modelling [8].

When new concepts are introduced into the conversation, an open domain dialogue manager must be able to transfer the knowledge of existing concepts and adapt its behaviour using a small amount of training data. In [9] a method which supports domain extension to cover a single new concept is described, incorporating the ideas of *transfer learning* [10]. Here we reformulate this technique to allow repeated incremental domain extension so that the range of concepts that the dialogue system can converse about grows over time. We demonstrate that a statistical spoken dialogue manager providing restaurant information for San Francisco can double its slot coverage, using only 1000 adaptation dialogues with real users. We also show that this adaptation technique can be used to improve policies bootstrapped using a user simulator.

The rest of the paper is organised as follows. First, we review dialogue management optimisation based on a Gaussian process model in a partially observable Markov decision process (POMDP) setting. Then, we formulate an incremental policy adaptation scheme. Following that, we explain the experimental set-up, consisting of the Bayesian Update of Dialogue State dialogue manager (Section 4.1), which operates in a domain described by a set of ontologies automatically derived from the web (Section 4.3). The experiments are conducted using subjects recruited via the Amazon Mechanical Turk crowdsourcing service (Section 4.5). In Section 3 we present results obtained during adaptation and in Section 6 we give the final evaluation results. Section 7 presents conclusions and future research directions.

2. Gaussian processes in POMDPs

The input to the dialogue manager is typically an N-best list of scored hypotheses obtained from the spoken language understanding unit. Based on this input, at every dialogue turn, a POMDP dialogue manager maintains a distribution of possible dialogue states called the *belief state*, $\mathbf{b} \in \mathcal{B}$. The role of a dialogue policy π is to map the belief state $\mathbf{b}$ into a system action $a \in \mathcal{A}$ so as to maximise the expected cumulative reward which is a measure of dialogue quality.

The expected cumulative reward is defined by the Q-function as:

$$Q(\mathbf{b}, a) = E_{\pi}\left(\sum_{\tau=t+1}^{T} \gamma^{\tau-t-1} r_{\tau} | b_t = \mathbf{b}, a_t = a\right), \quad (1)$$

where r_τ is the immediate reward obtained at time τ, T is the dialogue length and γ is the discount factor, $0 < \gamma \leq 1$. Optimising the Q-function is then equivalent to optimising the policy π.

GP-Sarsa is an on-line reinforcement learning algorithm that models the Q-function as a Gaussian process [11], $Q(\mathbf{b}, a) \sim \mathcal{GP}\left(0, k((\mathbf{b}, a), (\mathbf{b}, a))\right)$ where the kernel $k(\cdot, \cdot)$ is factored into separate kernels over the belief state and action space $k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}')k_{\mathcal{A}}(a, a')$. For a sequence of belief state-action pairs $\mathbf{B} = [(\mathbf{b}^0, a^0), \ldots, (\mathbf{b}^t, a^t)]^\intercal$ visited in training dialogues and the corresponding observed immediate rewards $\mathbf{r} = [r^1, \ldots, r^t]^\intercal$, the posterior of the Q-function for any belief state-action pair $(\mathbf{b}, a)$ is defined by the following:

$$\begin{aligned}
&Q(\mathbf{b}, a)|\mathbf{r}, \mathbf{B} \sim \mathcal{N}(\overline{Q}(\mathbf{b}, a), cov((\mathbf{b}, a), (\mathbf{b}, a))),\\
&\overline{Q}(\mathbf{b}, a) = \mathbf{k}(\mathbf{b}, a)^\intercal \mathbf{H}^\intercal (\mathbf{H}\mathbf{K}\mathbf{H}^\intercal + \sigma^2 \mathbf{H}\mathbf{H}^\intercal)^{-1} \mathbf{r},\\
&cov((\mathbf{b}, a), (\mathbf{b}, a)) = k((\mathbf{b}, a), (\mathbf{b}, a)) -\\
&\mathbf{k}(\mathbf{b}, a)^\intercal \mathbf{H}^\intercal (\mathbf{H}\mathbf{K}\mathbf{H}^\intercal + \sigma^2 \mathbf{H}\mathbf{H}^\intercal)^{-1} \mathbf{H}\mathbf{k}(\mathbf{b}, a),\\
&\mathbf{k}(\mathbf{b}, a) = [k((\mathbf{b}^0, a^0), (\mathbf{b}, a)), \ldots, k((\mathbf{b}^t, a^t), (\mathbf{b}, a))]^\intercal,\\
&\mathbf{K} = [\mathbf{k}((\mathbf{b}^0, a^0)), \ldots, \mathbf{k}((\mathbf{b}^t, a^t))],
\end{aligned} \quad (2)$$

We would like to thank Nesrine Ben Mustapha for providing the ontologies used here. The research leading to this work was funded by the EC FP7 programme FP7/2011-14 under grant agreement no. 287615 (PARLANCE).

where $\mathbf{K}$ is the Gram matrix – the matrix of the kernel function values for visited points $\mathbf{B}$, σ^2 is an additive noise parameter which controls how much variability in the Q-function estimate we expect during the process of learning and $\mathbf{H}$ is a linear operator that captures the reward lookahead from the Q-function (see Eq. 1) given by

$$\mathbf{H} = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix}. \tag{3}$$

A more detailed explanation of the use of GP-Sarsa in dialogue systems is given in [12].

3. Incremental policy adaptation

There are two distinct cases to consider when performing adaptation. The first is the case when the dynamics of the environment change, for example if the user changes. The second is the situation where the belief and the action space are different because the dialogue domain changes.

3.1. Adaptation without changes in the belief and action space

When the dynamics of the environment change, the system actions do not lead to the same cumulative rewards (Eq 1) and therefore the underlying Q-function changes. In this case, we can use the current estimate of the Q-function to find the adapted Q-function in the following way.

If we assume that the Gaussian process places a prior mean $m(\mathbf{b}, a)$ on the Q-function, $Q(\mathbf{b}, a) \sim \mathcal{GP}\left(m(\mathbf{b}, a), k((\mathbf{b}, a), (\mathbf{b}, a))\right)$ then the posterior mean $\overline{Q}(\mathbf{b}, a)$ is given by [13]:

$$\overline{Q}(\mathbf{b}, a) = m(\mathbf{b}, a) + \mathbf{k}(\mathbf{b}, a)^{\mathsf{T}}\mathbf{H}^{\mathsf{T}}(\mathbf{H}\mathbf{K}\mathbf{H}^{\mathsf{T}} + \sigma^2\mathbf{H}\mathbf{H}^{\mathsf{T}})^{-1}(\mathbf{r} - \mathbf{m}), \tag{4}$$

where $\mathbf{m} = [m(\mathbf{b}^0, a^0), \ldots, m(\mathbf{b}^t, a^t)]^{\mathsf{T}}$. The estimate of the variance is same as in Eq. 2. The mean of the Gaussian process posterior (Eq. 4) defines a function of $(\mathbf{b}, a)$ and can thus be used as the prior mean $m(\mathbf{b}, a)$ for another Gaussian process as demonstrated in [9].

This can be even further extended. The resulting mean of the Gaussian process posterior from Eq. 4 is again a function of $(\mathbf{b}, a)$ and can be used as a prior for yet another Gaussian process. In that way, every time the underlying function changes the current estimate of the Gaussian process posterior can be used as the prior mean and any additional training data is then used to estimate the posterior. This formulation can be implemented in a recursive manner where each Gaussian process has an associated Gaussian process that recursively defines its prior.

More formally, consider a sequence of Gaussian process posteriors $Q_1(\mathbf{b}, a)|\mathbf{r}_1, \mathbf{B}_1$, $Q_2(\mathbf{b}, a)|\mathbf{r}_2, \mathbf{B}_2$, $\ldots$, $Q_l(\mathbf{b}, a)|\mathbf{r}_l, \mathbf{B}_l$ such that the prior mean of the j^{th} Gaussian process is the posterior mean of the $(j-1)^{th}$ Gaussian process and the prior mean for Gaussian process Q_1 is zero. Then, the posterior mean of $Q_l(\mathbf{b}, a)|\mathbf{r}_l, \mathbf{B}_l$ can be expressed as

$$\begin{aligned} &\overline{Q}_l(\mathbf{b}, a) = \mathbf{k}_1(\mathbf{b}, a)^{\mathsf{T}}\mathbf{H}_1^{\mathsf{T}}(\mathbf{H}_1\mathbf{K}_1\mathbf{H}_1^{\mathsf{T}} + \sigma^2\mathbf{H}_1\mathbf{H}_1^{\mathsf{T}})^{-1}\mathbf{r}_1 + \\ &+\mathbf{k}_2(\mathbf{b}, a)^{\mathsf{T}}\mathbf{H}_2^{\mathsf{T}}(\mathbf{H}_2\mathbf{K}_2\mathbf{H}_2^{\mathsf{T}} + \sigma^2\mathbf{H}_2\mathbf{H}_2^{\mathsf{T}})^{-1}(\mathbf{r}_2 - \mathbf{m}_2) \\ &+ \ldots \\ &+\mathbf{k}_l(\mathbf{b}, a)^{\mathsf{T}}\mathbf{H}_l^{\mathsf{T}}(\mathbf{H}_l\mathbf{K}_l\mathbf{H}_l^{\mathsf{T}} + \sigma^2\mathbf{H}_l\mathbf{H}_l^{\mathsf{T}})^{-1}(\mathbf{r}_l - \mathbf{m}_l), \end{aligned} \tag{5}$$

where $\mathbf{m}_j = [\overline{Q}_{j-1}(\mathbf{b}^0, a^0), \ldots, \overline{Q}_{j-1}(\mathbf{b}^t, a^t)]^{\mathsf{T}}, j = 2, \ldots, l$.

3.2. Adaptation with incremental changes in the belief and action space

Assume now that the domain incrementally changes, i.e. that each subsequent belief state space $\mathcal{B}_j$ has additional dimensions compared to the previous belief state space $\mathcal{B}_{j-1}$ and that every subsequent action space is a superset of the previous action space, $\mathcal{A}_{j-1} \subset \mathcal{A}_j, j = 2, \ldots, l$. In order to apply adaptation as derived in Eq. 5 one needs to be able to calculate vectors $\mathbf{k}_1(\mathbf{b}, a), \ldots, \mathbf{k}_l(\mathbf{b}, a)$ for a point $(\mathbf{b}, a) \in \mathcal{B}_l \times \mathcal{A}_l$. From the definition in Eq. 2 we can see that $\mathbf{k}_j(\mathbf{b}, a)$ is a vector of kernel values for points visited during learning for that particular domain $\mathbf{k_j}(\mathbf{b}, a) = [k((\mathbf{b}_j^0, a_j^0), (\mathbf{b}, a)), \ldots, k((\mathbf{b}_j^t, a_j^t), (\mathbf{b}, a))]^{\mathsf{T}}, j = 1, \ldots, l$. Therefore, in principle, one needs to be able to calculate the kernel function $k((\mathbf{b}', a'), (\mathbf{b}, a))$ where $(\mathbf{b}', a') \in \mathcal{B}_j \times \mathcal{A}_j$, $(\mathbf{b}, a) \in \mathcal{B}_i \times \mathcal{A}_i$ and $j \leq i$. Since the kernel function is factored into the kernel for the belief state space and the action space, this needs to be done separately for each of these spaces. One way of doing this for the belief state space is to only consider the common dimensions between $\mathcal{B}_i$ and $\mathcal{B}_j$. In terms of the action space the kernel function is defined only on the actions that appear both in $\mathcal{A}_i$ and $\mathcal{A}_j$ and set to 0 otherwise.

4. Experimental Set-up

To investigate the effectiveness of the incremental policy adaptation scheme described above, the Bayesian Update of Dialogue State dialogue manager was adapted in interaction with real users. A user simulator was deployed for training the initial policies.

4.1. BUDS dialogue manager

The Bayesian Update of Dialogue State (BUDS) dialogue manager is a POMDP-based dialogue manager [5] which factorises the dialogue state into conditionally dependent elements, arranged into a dynamic Bayesian network. Thus, the belief state consists of the marginal posterior probability distribution over hidden nodes in the Bayesian network. The hidden nodes consist of the history nodes and the goal nodes for each concept.

To apply GP policy optimisation, a kernel function must be defined on both the belief state space $\mathcal{B}$ and the action space $\mathcal{A}$. In this case, the same approach is taken as in [9]. The kernel function on the belief state space is constructed from the sum of individual kernels over the hidden node distributions, such that the kernel function of two corresponding nodes is based on the expected likelihood kernel [14], which is a simple linear inner product:

$$k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}') = \sum_h \langle \mathbf{b}_h, \mathbf{b}_h' \rangle, \tag{6}$$

where $\mathbf{b}_h$ is the probability distribution encoded in the h^{th} hidden node. This kernel gives the expectation of one belief state distribution under the other.

For history nodes, the kernel is a simple inner product between the corresponding node distributions.The kernel over two goal nodes is calculated as the dot product of vectors, where each vector represents the corresponding distribution sorted into order of probability.

For the action space kernel, the δ-kernel is used defined by:

$$k_{\mathcal{A}}(a, a') = \delta_a(a'), \tag{7}$$

where $\delta_a(a') = 1$ iff $a = a'$, 0 otherwise.

4.2. The agenda-based user simulator

For training the initial policies, the agenda-based user simulator was used, which factorises the user state into an *agenda* and a *goal* to ensure consistent, goal-directed behaviour [15, 16].

A dialogue was considered successful if the system provided all the information that the user asked for. The reward function was set to give a reward of 20 for successful dialogues, zero otherwise. In addition, 1 was deducted for each dialogue turn to encourage shorter dialogues.

4.3. Ontologies

The evaluation system provides restaurant information for San Francisco. The core system understands just 3 slot types: *food*, *area* and *pricerange*. Extended systems successively learn to handle three new slots: *near*, *allowedforkids* and *goodformeal*. The corresponding domains are described by a set of ontologies that are automatically generated using information from the web [17], see Table 1.

Ontology	Attributes (# of vaues)
SFCore	*food*(59), *area*(155), *pricerange*(3)
SF1Ext	SFCore + *near*(39)
SF2Ext	SF1Ext + *allowedforkids*(2)
SF3Ext	SF2Ext +*goodformeal*(4)

Table 1: Expanding domains

The three domain extensions have attributes of very different nature. Attribute *near* has a large number of possible values, while *allowedforkids* is a binary attribute and *goodformeal* has only a few values such as *dinner*, *lunch* etc. Therefore, every time a new attribute is added the dialogue manager must adjust its behaviour. The overall number of entities is 239 and the user can enquire about *phone*, *address*, *postcode* and *prices*.

4.4. Extended belief state

Every time the domain is extended with a new attribute, a set of hidden nodes is added to the Bayesian network. The dialogue manager then needs to be able to update the belief state of this larger network. In order to do that, the similarity is computed between the new attribute and each existing attribute by comparing cardinalities. Then, the transition probabilities for the new nodes are defined as the transition probabilities for the nodes that correspond to the most similar attribute.

In order to adapt the policy to new domains one needs to define the kernel function between the belief states that come from different domains, where one is the extension of the other. We only consider attributes that are the same in both domains:

$$k_{\mathcal{B}}(\mathbf{b}^{\mathsf{B}}, \mathbf{b}^{\mathsf{E}}) = \sum_{h \in \mathsf{B}} \langle \mathbf{b}_h^{\mathsf{B}}, \mathbf{b}_h^{\mathsf{E}} \rangle, \tag{8}$$

where h are the hidden nodes in the basic domain B. The kernel function between two sets of actions is

$$k_{\mathcal{A}}(a^{\mathsf{B}}, a^{\mathsf{E}}) = \delta_{a^{\mathsf{B}}}(a^{\mathsf{E}}), \tag{9}$$

where $a^{\mathsf{E}} \in \mathcal{A}^{\mathsf{B}}$.

4.5. Crowd-sourcing via Amazon MTurk

In order to adapt and evaluate policies with humans, we integrated the BUDS dialogue manager into a telephone-based spoken dialogue system and recruited the subjects via the Amazon Mechanical Turk service in a set-up similar to [18, 6]. We collected a small amount of text dialogues using a handcrafted policy operating on SF3Ext domain and used them to train the language model. The spoken language understanding component is a variant of Phoenix parser and the language generation unit is template-based.

The MTurk users were assigned specific tasks in different domains. They were asked to find restaurants that have particular features as defined by the given task. After each dialogue the users were asked whether they judged the dialogue to be successful or not. Based on that binary rating, the subjective success was calculated as well as the average reward. An objective rating was also computed by comparing the system outputs with the predefined task. During adaptation only dialogues where both objective and subjective score were the same were used, while for evaluation all dialogues were taken into account.

5. Adaptation in interaction with real users

We investigated two cases of adaptation – when the environment changes and when the domain changes incrementally. Rather than trying to expand the full domain immediately, we gradually make the task more difficult, adopting the ideas of *curriculum learning* [19]. The adaptation schedule is illustrated in Fig. 1. First a policy for the SFCore domain was trained using the user simulator and learning on a large number of dialogues until the policy converged. For comparison, a policy for the full SF3Ext domain was also trained on the simulator.

The SFCore policy trained on the user simulator was then used as a prior to obtain an SFCore policy adapted using real users. In this case the domain did not change, only the environment, so we used the technique described in Section 3.1 using 200 adaptation dialogues. Then, the resulting policy was taken as a prior for the SF1Ext domain and adapted using a further 300 dialogues. The domain was then again extended to SF2Ext and 300 adaptation dialogues were performed. The final extension to the SF3Ext domain had 200 adaptation dialogues[1].

For comparison, we also adapted the SF3Ext policy trained on the user simulator using 300 dialogues with real users, c.f. Fig. 1.

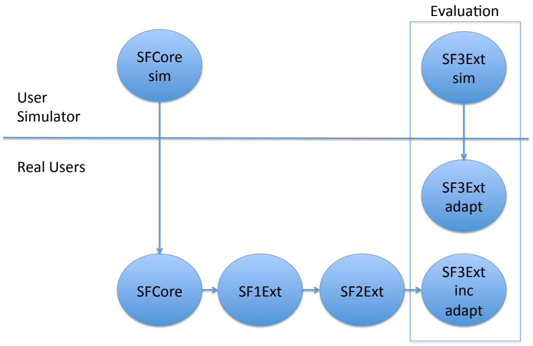

Figure 1: Adaptation schedule

For each adaptation experiment we calculated the moving average reward using a moving window of 150 dialogues. The results are shown in Figs. 2,3,4 and 5. The shaded area represents one standard error. Bearing in mind that the initial parts of

[1] The performance deteriorated after 200 dialogues due to an increase in speech understanding errors.

the graphs are not indicative of the performance since the number of dialogues in the beginning is very small, Figs. 2,4 and 5 show an upward trend in performance.

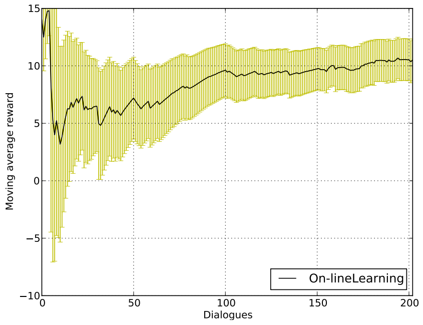

Figure 2: Adaptation to real users on SFCore domain

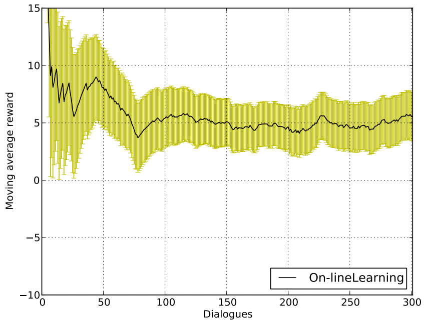

Figure 3: Adaptation to SF1Ext domain

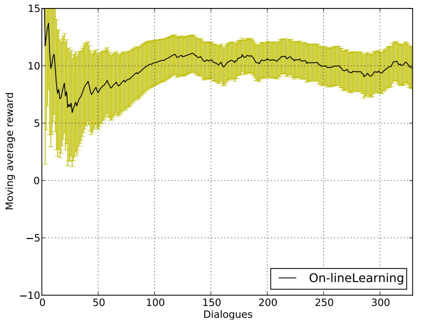

Figure 4: Adaptation to SF2Ext domain

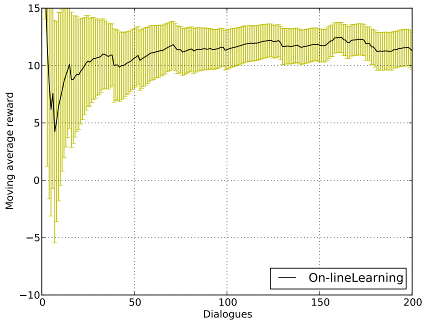

Figure 5: Adaptation to SF3Ext domain

The performance of the SF1Ext domain (Fig 3) however is clearly worse than for the other domains. We transcribed the collected dialogues and calculated the word error rate (WER) for each domain extension and found that it is considerably larger for SF1Ext (Table 2).

Table 2: Word error rate for different domains

Domain	#Adaptation diags	#Diags	WER
SFCore	200	399	15
SF1Ext	300	602	22
SF2Ext	329	534	19
SF3Ext	200	320	18

It is somewhat counter-intuitive that the word error rate for a larger domain is lower than for a smaller domain. The reason for this may be that the larger system can use attributes which are easy to recognise such as *allowedforkids* and *goodformeal* to narrow down the user's request without always needing to specify attribute *near* which, due to the very large number of possible values, proves particularly challenging for speech understanding.

6. Evaluation

To investigate the effectiveness of the adaptation technique, we evaluated three policies in direct interaction with real users: the policy trained on the simulator for the SF3Ext domain (sim), the policy adapted to SF3Ext using the policy trained on the user simulator as the prior (adapt), the policy incrementally adapted to SF3Ext starting from the SFCore policy trained on the user simulator (inc adapt). The results are given in Table 3 along with one standard error. The results show that the SF3Ext adapt policy is significantly better than the SF3Ext sim policy and that there is no statistical difference between the SF3Ext inc adapt policy and the SF3Ext adapt policy.

Table 3: Evaluation of policies operating on the SF3Ext domain

Policy	#Diags	Reward	Success(%)	#Turns
sim	306	9.2 ± 0.6	80.7 ± 2.3	6.9 ± 0.3
adapt	311	10.5 ± 0.5	83.9 ± 2.1	6.3 ± 0.2
inc adapt	305	10.3 ± 0.5	82.3 ± 2.2	6.1 ± 0.2

7. Conclusions

These results have two important implications. First, a policy trained on a user simulator can be improved in interaction with real users. In this case, the domain is the same but the environment has changed. By using additional adaptation dialogues, the policy captures changes in the environment while reusing the knowledge of the policy trained on the user simulator.

Secondly, this form of adaptation appears to be practical even when the domain significantly expands. A policy can be trained for a limited domain and then incrementally adapted to a larger domain without supervision in direct interaction with real users using a small number of adaptation dialogues and without the expense of building a user simulator for the extended domain.

The ability to handle an expanding domain represents a clear step towards open domain dialogue. The next step is to show that policies trained for different domains can be combined into some form of distributed policy representation which enables a dialogue to seamlessly switch from one topic to another. This will be the focus of future work.

8. References

[1] N. Roy, J. Pineau, and S. Thrun, “Spoken dialogue management using probabilistic reasoning,” in *Proceedings of ACL*, 2000.

[2] B. Zhang, Q. Cai, J. Mao, E. Chang, and B. Guo, “Spoken Dialogue Management as Planning and Acting under Uncertainty,” in *Proceedings of Eurospeech*, 2001.

[3] J. Williams and S. Young, “Partially Observable Markov Decision Processes for Spoken Dialog Systems,” *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[4] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management,” *Computer Speech and Language*, vol. 24, no. 2, pp. 150–174, 2010.

[5] B. Thomson and S. Young, “Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems,” *Computer Speech and Language*, vol. 24, no. 4, pp. 562–588, 2010.

[6] M. Gašić, C. Breslin, M. Henderson, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young, “On-line policy optimisation of Bayesian Dialogue Systems by human interaction,” in *Proceedings of ICASSP*, 2013.

[7] J. Williams, “Multi-domain learning and generalization in dialog state tracking,” in *Proceedings of SIGDIAL*, 2013.

[8] D. Litman and S. Pan, “Designing and evaluating an adaptive spoken dialogue system,” *User Modelling and User-Adapted Interaction*, vol. 12, pp. 111–137, 2002.

[9] M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young, “POMDP-based dialogue manager adaptation to extended domains,” in *Proceedings of SIGDIAL*, 2013.

[10] M. Taylor, P. Stone, and Y. Liu, “Transfer learning via inter-task mappings for temporal difference learning,” *J. Mach. Learn. Res.*, vol. 8, pp. 2125–2167, Dec. 2007.

[11] Y. Engel, S. Mannor, and R. Meir, “Reinforcement learning with Gaussian processes,” in *Proceedings of ICML*, 2005.

[12] M. Gašić and S. Young, “Gaussian Processes for POMDP-Based Dialogue Manager Optimization,” *TASSP*, vol. 22, no. 1, 2014.

[13] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: MIT Press, 2005.

[14] T. Jebara, R. Kondor, and A. Howard, “Probability product kernels,” *J. Mach. Learn. Res.*, vol. 5, pp. 819–844, Dec. 2004.

[15] J. Schatzmann, “Statistical User and Error Modelling for Spoken Dialogue Systems,” Ph.D. dissertation, University of Cambridge, 2008.

[16] S. Keizer, M. Gašić, F. Jurčíček, F. Mairesse, B. Thomson, K. Yu, and S. Young, “Parameter estimation for agenda-based user simulation,” in *Proceedings of SIGDIAL*, 2010.

[17] N. Ben Mustapha, M. Aufaure, H. Baazaoui-Zghal, and H. Ben Ghezala, “Query-driven approach of contextual ontology module learning using web snippets,” *Special issue on Database Management and Information Retrieval, Journal of Intelligent Information Systems*, 2013.

[18] F. Jurčíček, S. Keizer, M. Gašić, F. Mairesse, B. Thomson, K. Yu, and S. Young, “Real user evaluation of spoken dialogue systems using Amazon Mechanical Turk,” in *Proceedings of Interspeech*, 2011.

[19] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.