DISTRIBUTED DIALOGUE POLICIES FOR MULTI-DOMAIN STATISTICAL DIALOGUE MANAGEMENT

M. Gašić, D. Kim, P. Tsiakoulis and S. Young

Cambridge University Engineering Department

ABSTRACT

Statistical dialogue systems offer the potential to reduce costs by learning policies automatically on-line, but are not designed to scale to large open-domains. This paper proposes a hierarchical distributed dialogue architecture in which policies are organised in a class hierarchy aligned to an underlying knowledge graph. This allows a system to be deployed using a modest amount of data to train a small set of generic policies. As further data is collected, generic policies can be adapted to give in-domain performance. Using Gaussian process-based reinforcement learning, it is shown that within this framework generic policies can be constructed which provide acceptable user performance, and better performance than can be obtained using under-trained domain specific policies. It is also shown that as sufficient in-domain data becomes available, it is possible to seamlessly improve performance, without subjecting users to unacceptable behaviour during the adaptation period and without limiting the final performance compared to policies trained from scratch.[1]

Index Terms— open-domain, multi-domain, dialogue systems, POMDP, Gaussian process

1. INTRODUCTION

Statistical spoken dialogue systems allow policies to be learned directly from data thereby avoiding the need for hand-crafting dialogue decisions [1, 2]. Furthermore, the recent introduction of sample-efficient reinforcement learning algorithms has substantially reduced the number of training dialogues required to train effective policies [3, 4] and this has enabled systems to be trained on-line in direct interaction with human users [5].

Current approaches to statistical dialogue assume that all possible dialogue states can be encapsulated in a single real-valued vector referred to as a *belief state* $\boldsymbol{b}$ and that dialogue decisions can be defined by a *policy* which maps belief states into actions $\pi(\boldsymbol{b}) \rightarrow a$. This model works well for applications in specific limited domains and it is sufficiently flexible to allow a specific domain to be adapted and extended on-the-fly in direct interaction with human users [6]. However, it does not readily scale to support multiple domains in which the user wishes to move from one topic to another.

In order to support large and potentially open domains, techniques that can reuse existing knowledge and adapt on-line are needed. Such a system has extensive knowledge which could be structured in the form of a hierarchical ontology populated with instances of the various types of entity that it knows about. Following Google, such data structures are frequently referred to as *Knowledge Graphs (KGs)* and they have already been explored in connection with spoken language understanding tasks [7, 8]. For the purposes of this paper, we consider a KG to be a tree-structured class hierarchy in which each class represents a domain or topic by a set of properties (slot-value pairs), and derived classes represent specialisations of that domain that inherit slots from their parent and add additional slots specific to the specialisation. For example (see Fig 1), a generic venue class might define the common properties of a venue such as name, area, phone, etc. A derived class such as restaurant will then inherit the common properties of a venue, but add restaurant-specific slots such as type of food, whether kids-allowed, etc. Actual entities are then class instances in which each slot has a specific value. The goal of a typical dialogue might then be to identify specific instances which match the user's constraints (e.g. "Find a restaurant nearby serving Italian food").

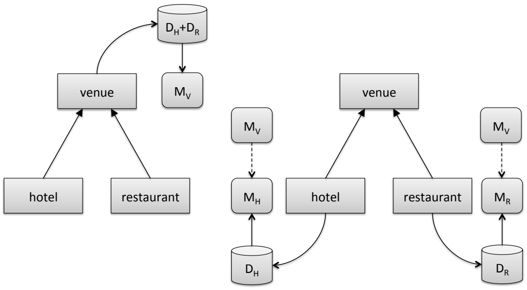

Fig. 1. Training a generic venue policy model M_V on data pooled from two subdomains $D_R + D_H$ (left); and training specific policy models M_R and M_H using the generic policy M_V as a prior and additional in-domain training data (right).

One way to build a dialogue manager which can operate across a large knowledge graph is to decompose the dialogue policy into a set of topic specific policies that are distributed across the class nodes in the graph. Initially, there will be relatively little training data and the system will need to rely on generic policies attached to high level generic class nodes which have been trained on whatever examples are available from across the pool of derived classes. As more data is collected, specific policies can be trained for each derived class[2]. An example of this is illustrated in Fig 1. On the left side is the initial situation where conversations about hotels and restaurants are conducted using a generic model M_V trained on example dialogues from both the hotel and restaurant domains. Once the system has been deployed and more training data has been collected, specific

We would like to thank Nesrine Ben Mustapha for providing the ontologies used here. This work was partly supported by PARLANCE (www.parlance-project.eu), an EU Seventh Framework Programme project (grant number 287615).

[1] *Note: due to an error in the configuration settings, some of the results presented in the original version of this paper in the proceedings were not directly comparable. This version of the paper has the corrected results.*

[2] cf analogy with speech recognition adaptation using regression trees[9]

restaurant and hotel models M_R and M_H can be trained.[3]

This type of multi-domain model assumes an agile deployment strategy which can be succinctly described as “deploy, collect data, and refine”. Its viability depends on the following assumptions:

1. it is possible to construct generic policies which provide acceptable user performance across a range of differing domains;
2. as sufficient in-domain data becomes available, it is possible to seamlessly adapt the policy to improve performance, without subjecting users to unacceptable disruptions in performance during the adaptation period.

The aim of this paper is to investigate the validity of these assumptions.

The remainder of the paper is organised as follows. In Section 2, the use of Gaussian process-based reinforcement learning (GPRL) is briefly reviewed. The key advantage of GPRL in this context is that in addition to being data efficient, it directly supports the use of an existing model as a prior thereby facilitating incremental adaptation. In Section 3, various strategies for building a generic policy are considered and evaluated. In Section 4, sub-domain policy adaptation is examined using the associated generic policy as a prior. Results are then presented for live on-line adaptation with human users in Section 5. Finally, in Section 6, conclusions are presented.

2. GP-BASED REINFORCEMENT LEARNING

The input to a statistical 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 distribution of possible dialogue states called the *belief state*, an element of *belief space* $\boldsymbol{b} \in \mathcal{B}$, is estimated. The quality of a dialogue is defined by a *reward function* $r(\boldsymbol{b}, a)$ and the role of a dialogue policy π is to map the belief state $\boldsymbol{b}$ into a system action, an element of *action space* $a \in \mathcal{A}$, at each turn so as to maximise the expected cumulative reward.

The expected cumulative reward for a given belief state $\boldsymbol{b}$ and action a is defined by the Q-function:

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

where r_τ is the immediate reward obtained at time τ, T is the dialogue length and γ is a 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 [10]:

$$Q(\boldsymbol{b}, a) \sim \mathcal{GP}\left(m(\boldsymbol{b}, a), k((\boldsymbol{b}, a), (\boldsymbol{b}, a))\right) \qquad (2)$$

where $m(\boldsymbol{b}, a)$ is the prior mean and the kernel $k(\cdot, \cdot)$ is factored into separate kernels over belief and action spaces $k_{\mathcal{B}}(\boldsymbol{b}, \boldsymbol{b}')k_{\mathcal{A}}(a, a')$.

For a training sequence of belief state-action pairs $\boldsymbol{B} = [(\boldsymbol{b}^0, a^0), \ldots, (\boldsymbol{b}^t, a^t)]^\mathsf{T}$ and the corresponding observed immediate rewards $\boldsymbol{r} = [r^1, \ldots, r^t]^\mathsf{T}$, the posterior of the Q-function for any belief state-action pair $(\boldsymbol{b}, a)$ is given by:

$$Q(\boldsymbol{b}, a)|\boldsymbol{r}, \boldsymbol{B} \sim \mathcal{N}(\overline{Q}(\boldsymbol{b}, a), cov((\boldsymbol{b}, a), (\boldsymbol{b}, a))) \qquad (3)$$

where the posterior mean and covariance take the form:

$$\begin{array}{l} \overline{Q}(\boldsymbol{b}, a) = \boldsymbol{k}(\boldsymbol{b}, a)^\mathsf{T} \boldsymbol{H}^\mathsf{T} (\boldsymbol{HKH}^\mathsf{T} + \sigma^2 \boldsymbol{HH}^\mathsf{T})^{-1} (\boldsymbol{r} - \boldsymbol{m}), \\ cov((\boldsymbol{b}, a), (\boldsymbol{b}, a)) = k((\boldsymbol{b}, a), (\boldsymbol{b}, a)) - \\ \quad \boldsymbol{k}(\boldsymbol{b}, a)^\mathsf{T} \boldsymbol{H}^\mathsf{T} (\boldsymbol{HKH}^\mathsf{T} + \sigma^2 \boldsymbol{HH}^\mathsf{T})^{-1} \boldsymbol{Hk}(\boldsymbol{b}, a) \end{array} \qquad (4)$$

where $\boldsymbol{k}(\boldsymbol{b}, a) = [k((\boldsymbol{b}^0, a^0), (\boldsymbol{b}, a)), \ldots, k((\boldsymbol{b}^t, a^t), (\boldsymbol{b}, a))]^\mathsf{T}$, $\boldsymbol{K}$ is the Gram matrix [11], $\boldsymbol{H}$ is a band matrix with diagonal $[1, -\gamma]$, $\boldsymbol{m} = [m(\boldsymbol{b}^0, a^0), \ldots, m(\boldsymbol{b}^t, a^t)]^\mathsf{T}$ and σ^2 is an additive noise factor which controls how much variability in the Q-function estimate is expected during the learning process. Function $m()$ is 0 for the initial GP model estimation, and is subsequently set to the posterior mean of the previous GP model during incremental adaptation (see [5, 6] for details).

To use GPRL for dialogue, a kernel function must be defined on both the belief state space $\mathcal{B}$ and the action space $\mathcal{A}$. Here we use the Bayesian Update of Dialogue State (BUDS) dialogue model [12] and we use the same approach as in [13] for defining the kernel function. The action space consists of a set of slot-dependent and slot-independent summary actions which are mapped to master actions using a set of rules and the kernel is defined as:

$$k_{\mathcal{A}}(a, a') = \delta_a(a') \qquad (5)$$

where $\delta_a(a') = 1$ iff $a = a'$, 0 otherwise. The belief state consists of the probability distributions over the Bayesian network hidden nodes that relate to the dialogue history for each slot and the value of each user goal slot. The dialogue history nodes can take a fixed number of values, whereas user goals range over the values defined for that particular slot in the ontology and can have very high cardinalities. User goal distributions are therefore sorted according to the probability assigned to each value since the choice of summary action does not depend on the values but rather on the overall shape of each distribution. The kernel function over both dialogue history and user goal nodes is based on the expected likelihood kernel [14], which is a simple linear inner product. The kernel function for belief space is then the sum over all the individual hidden node kernels:

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

where $\boldsymbol{b}_h$ is the probability distribution encoded in the h^{th} hidden node. The dimensionality of the belief space grows with the number of slots and the cardinality of each. The impact of the latter can be mitigated by truncating the values to just the k most probable [12]. The impact of a large and growing number of slots requires some form of partitioning of belief space. The distributed policy design proposed in this paper is one approach to dealing with this.

3. DESIGNING A GENERIC POLICY

To investigate the distributed policy model described in Section 1, dialogue policies were built for restaurants and hotels in San Francisco, USA, referred to as SFRestaurant and SFHotel respectively. SFRestaurant contains 239 entities and SFHotel has 182 entites. These subdomains are described by ontologies that were automatically generated using information from the web [15]. A description of slots and the values that they take is given in Table 1, where bold identifies the goal slots that can be specified by the user and the remaining slots are informational slots that the user can query.

In GPRL, the computation of $Q(\boldsymbol{b}, a)$ requires the kernel function to be evaluated between $(\boldsymbol{b}, a)$ and each of the belief-action points in the training data. If the training data consists of dialogues from subdomains (restaurants and hotels in this case) which have

[3] Here a model M is assumed to include input mappings for speech understanding, a dialogue policy π and output mappings for generation. In this paper, we are only concerned with dialogue management and hence the dialogue policy component π of each model.

Table 1. Slots and their cardinalities for the two subdomains. Bold indicates slots that can be specified by the user to constrain a search, the remainder are informable slots. The stars indicate abstract slots in the abstract and renorm styles of generic policy.

	SFRestaurant		SFHotel	
	Slot	#Card	Slot	#Card
	name	239	**name**	182
	area	155	**area**	155
	near	39	**near**	28
	pricerange	3	**pricerange**	4
*	**food**	59	**dogsallowed**	2
*	**goodformeal**	4	**hasinternet**	2
*	**kids-allowed**	2	**acceptscards**	2
	price	96	phone	177
	phone	240	addr	180
	addr	238	postcode	19
	postcode	21	-	-

domain-specific slots and actions, a strategy is needed for computing the kernel function between domains. Here three different strategies are considered.

The first and simplest is to consider only slots that are common to both subdomains $\mathcal{R}$ and $\mathcal{H}$:

$$k_{\mathcal{B}}(\boldsymbol{b}^{\mathcal{H}}, \boldsymbol{b}^{\mathcal{R}}) = \sum_{h \in \mathcal{H} \cap \mathcal{R}} \langle \boldsymbol{b}_h^{\mathcal{H}}, \boldsymbol{b}_h^{\mathcal{R}} \rangle, \tag{7}$$

When goal nodes are paired with differing cardinalities (eg name in Table 1), the shorter vector is padded with zeros. Similarly, the kernel over actions only considers actions that are the same in both subdomains and in other cases returns 0. This strategy is referred to as **mutual**. In the second strategy, non-matching slots are renamed as slot-1, slot-2, etc and treated as abstract slots so that they are the same in both subdomains, these are the starred rows in Table 1. Hence for example, food is matched with dogs allowed, and so on. As with the mutual case, when goal nodes are paired with differing cardinalities, the shorter vector is padded with zeros. This strategy is referred to as **abstract**. Finally, a variation of the abstract strategy is considered where, instead of padding the shorter goal vectors with zeros, the longer vectors are normalised to match the length of the shorter. We refer to this strategy as **renorm**. Adaptation strategies based on [16] are also possible but are reliant on increasing the dimensionality.

In order to investigate the effectiveness of these strategies, generic policies were trained and then tested in both subdomains. In all cases the training data consisted of an equal number of restaurant and hotel dialogues. In addition, in-domain policies were trained as a reference. Training and testing was performed on an agenda-based simulated user operating on the dialogue act level [17, 18]. The reward function allocates -1 at each turn to encourage shorter dialogues, plus 20 at the end of each successful dialogue. The user simulator includes an error generator and this was set to generate incorrect user inputs 15% of time. For each condition, 10 policies were trained using different random seeds using differing numbers of training dialogues. Each policy is then evaluated using 1000 dialogues on each subdomain. The overall average reward, success rate and number of turns is given in Table 2 together with a 95% confidence interval. The most important measure is the average reward, since the policies are trained to maximise this.

There are two important conclusions to be drawn from these results. First, all generic policies perform better than the in-domain policies trained only on the data available for that subdomain (i.e. half of the training data available for the generic policy in this case) and this is especially the case when training data is limited. Secondly, in the majority of cases the strategies using abstract slots provide the best performance.

Table 2. Comparison of strategies for training generic policies. In-domain performance is measured in terms of reward, success rate and the average number of turns per dialog. Results are given with a 95% confidence interval.

Strategy	#Dialogs	Reward	Success	#Turns
SFRestaurant				
in-domain	250	3.26 ± 0.21	60.02 ± 0.97	8.65 ± 0.08
in-domain	500	5.00 ± 0.21	68.17 ± 0.91	8.55 ± 0.07
mutual	500	3.88 ± 0.21	62.60 ± 0.95	8.56 ± 0.07
abstract	500	4.48 ± 0.21	67.35 ± 0.92	8.89 ± 0.08
renorm	500	4.49 ± 0.21	67.44 ± 0.92	8.89 ± 0.08
in-domain	2500	7.95 ± 0.17	83.02 ± 0.75	8.55 ± 0.07
in-domain	5000	8.68 ± 0.15	86.67 ± 0.67	8.54 ± 0.07
mutual	5000	8.28 ± 0.16	84.63 ± 0.71	8.52 ± 0.07
abstract	5000	8.58 ± 0.15	86.21 ± 0.68	8.52 ± 0.07
renorm	5000	8.60 ± 0.16	86.32 ± 0.68	8.56 ± 0.07
SFHotel				
in-domain	250	3.58 ± 0.21	62.07 ± 0.96	8.75 ± 0.07
in-domain	500	4.83 ± 0.21	69.08 ± 0.92	8.89 ± 0.08
mutual	500	4.81 ± 0.21	68.00 ± 0.92	8.71 ± 0.08
abstract	500	5.27 ± 0.20	70.01 ± 0.90	8.64 ± 0.07
renorm	500	4.64 ± 0.21	68.59 ± 0.92	8.96 ± 0.08
in-domain	2500	8.40 ± 0.16	84.90 ± 0.71	8.46 ± 0.06
in-domain	5000	8.92 ± 0.15	87.48 ± 0.65	8.45 ± 0.06
mutual	5000	8.69 ± 0.15	85.83 ± 0.68	8.36 ± 0.06
abstract	5000	8.89 ± 0.15	87.19 ± 0.66	8.44 ± 0.06
renorm	5000	8.85 ± 0.15	87.45 ± 0.65	8.55 ± 0.07

4. ADAPTATION OF IN-DOMAIN POLICIES USING A GENERIC POLICY AS A PRIOR

We now investigate the effectiveness of a generic policy as a prior for training an in-domain policy as in the right hand side of Fig. 1. Since the strategy of using abstract slots provided the best overall generic performance and renorm strategy gave comparable performance, only abstract generic policies are used from now on. In order to examine the best and worst case, the abstract generic priors (from the 10 randomly seeded examples) that gave the best performance and the worst performance on each sub-domain trained with 500 and 5000 dialogues were selected. This results in four prior policies for each subdomain: abstract-500-worst, abstract-500-best, abstract-5000-worst and abstract-5000-best. In addition, a policy with no prior was also trained for each subdomain (i.e. the policy was trained from scratch). After every 5000 training dialogues each policy was evaluated with 1000 dialogues. The results are given in Fig. 2 and 3 with a 95%confidence interval. Performance at 0 training dialogues corresponds to using the generic policy as described in the previous section, or using a random policy for the no prior case.

The results from Figs. 2 and 3 demonstrate that the performance of the policy in the initial stages of learning are significantly improved using the generic policy as a prior, even if that prior is trained on a small number of dialogues and even if it was the worst performing prior from the batch of 10 training sessions. These results also show that the use of a generic prior does not limit the optimality of the final policy. In fact, the use of a prior can be seen as resetting the variance of a GP which could lead to better sample efficiency [19]. This may be the reason why in some cases the no-prior policies do

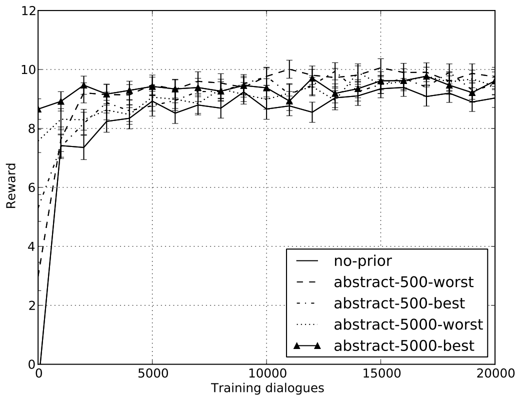

Fig. 2. Training policies with different priors – SFRestaurant

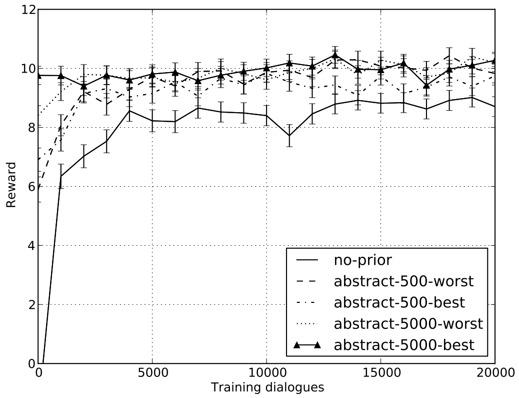

Fig. 3. Training policies with different priors – SFHotel

not catch up with the adapted policies.

In Table 3, the performance of the best performing generic prior is compared to the performance of the same policy adapted using an additional 50K dialogues. The results show that additional in-domain adaptation has the potential to improve the performance further. So when enough training data is available, it is beneficial to create individual in-domain policies rather than continuing to train the generic policy. The reason behind this may be that the optimal performance can only be reached when the training and the testing conditions match.

5. ADAPTATION IN INTERACTION WITH HUMAN USERS

In order to investigate how the use of a generic policy as a prior influences training in interaction with human users, a generic policy was adapted on-line in a real-time spoken dialogue system using subjects recruited via Amazon Mturk. Each user was assigned specific tasks in the SFRestaurant subdomain and then asked to call the system in a similar set-up to that described in [20, 21]. 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 assigned task specification. During training, only dialogues where both objective and subjective score were the same were used.

Two training schedules were performed in the SFRestaurant subdomain – one training from scratch without a prior and the other performing adaptation using the best generic prior obtained after 5000 training dialogues on the user simulator. For each training schedule three sessions were performed and the results were averaged to reduce any random variation. Fig. 4 shows the moving average reward as a function of the number of training dialogues. The moving window was set to 100 dialogues so that after the initial 100 dialogues each point on the graph is an average of 300 dialogues. The shaded area represents a 95% confidence interval. The initial parts of the graph exhibit more randomness in behaviour because the number of training dialogues is small.

Table 3. Performance of best generic prior when adapted using 50K additional dialogues. Results are given with one standard error.

Name	Rew	Suc	Tur
SFRestaurant			
best prior	8.66 ± 0.35	85.40 ± 2.19	8.32 ± 0.20
adapted	9.62 ± 0.30	89.60 ± 1.90	8.24 ± 0.19
SFHotel			
best prior	9.76 ± 0.31	88.80 ± 1.96	7.95 ± 0.21
adapted	10.27 ± 0.27	92.50 ± 1.64	8.20 ± 0.21

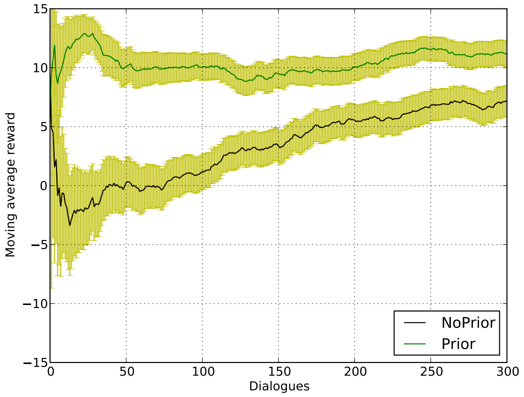

Fig. 4. Training in interaction with human users on SFRestaurant – moving average reward

The results show an upward trend in performance particularly for the policy that uses no prior. However, the performance obtained with the prior is significantly better than without a prior both in terms of the reward and the success rate. Equally importantly, unlike the system trained from scratch with no prior, the users of the adapted system are not subjected to poor performance during the early stages of training.

6. CONCLUSIONS

This paper has proposed a distributed multi-domain dialogue architecture in which dialogue policies are organised in a class hierarchy aligned to an underlying knowledge graph. The class hierarchy allows a system to be deployed by using a modest amount of data to train a small set of generic policies. As further data is collected, generic policies can be adapted to give in-domain performance. Using Gaussian process-based reinforcement learning, it has been shown that it is possible to construct generic policies which provide acceptable in-domain user performance, and better performance than can be obtained using under-trained domain specific policies. To construct a generic policy, a design consisting of all common slots plus a number of abstract slots which can be mapped to domain-specific slots works well. It has also been shown that as sufficient in-domain data becomes available, it is possible to seamlessly adapt to improve performance, without subjecting users to unacceptable disruptions in performance during the adaptation period and without limiting the final performance compared to policies trained from scratch. Future work will investigate the problem of finding adequate mappings for generic policies and training them with human users.

7. REFERENCES

[1] JD Williams and SJ Young, "Partially Observable Markov Decision Processes for Spoken Dialog Systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[2] SJ Young, M Gasic, B Thomson, and JD Williams, "Pomdp-based statistical spoken dialogue systems: a review," *Proceedings IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[3] M Gašić, F Jurčíček, S Keizer, F Mairesse, J Schatzmann, B Thomson, K Yu, and S Young, "Gaussian Processes for Fast Policy Optimisation of POMDP-based Dialogue Managers," in *Proceedings of SIGDIAL*, 2010.

[4] M Geist and O Pietquin, "Managing Uncertainty within the KTD Framework," in *Proceedings of the Workshop on Active Learning and Experimental Design*, Sardinia (Italy), 2011.

[5] M Gašić and S Young, "Gaussian Processes for POMDP-Based Dialogue Manager Optimization," *TASLP*, vol. 22, no. 1, 2014.

[6] M Gašić, D Kim, P Tsiakoulis, M Henderson, M Szummer, B Thomson, and S Young, "Incremental on-line adaptation of POMDP-based dialogue managers to extended domains," in *Proceedings of Interspeech*, 2014.

[7] Gökhan Tür, Minwoo Jeong, Ye-Yi Wang, Dilek Hakkani-Tür, and Larry P Heck, "Exploiting the semantic web for unsupervised natural language semantic parsing," in *Proceedings of Interspeech*, 2012.

[8] Larry P Heck, Dilek Hakkani-Tür, and Gökhan Tür, "Leveraging knowledge graphs for web-scale unsupervised semantic parsing.," in *Proceedings of Interspeech*, 2013, pp. 1594–1598.

[9] M.J F. Gales, "The Generation And Use Of Regression Class Trees For MLLR Adaptation," Tech. Rep. CUED/F-INFENG/TR.263, Cambridge University Engineering Dept, 1996.

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

[11] CE Rasmussen and CKI Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, Massachusetts, 2005.

[12] 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.

[13] 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.

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

[15] N Ben Mustapha, MA 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.

[16] 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.

[17] J Schatzmann, B Thomson, K Weilhammer, H Ye, and SJ Young, "Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System," in *Proceedings of HLT*, 2007.

[18] 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.

[19] Robert C. Grande, Thomas J. Walsh, and Jonathan P. How, "Sample efficient reinforcement learning with gaussian processes," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014, pp. 1332–1340.

[20] 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.

[21] 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.