Policy Learning for Domain Selection in an Extensible Multi-domain Spoken Dialogue System

Zhuoran Wang
Mathematical & Computer Sciences
Heriot-Watt University
Edinburgh, UK
zhuoran.wang@hw.ac.uk

Hongliang Chen, **Guanchun Wang**
Hao Tian, **Hua Wu**[†], **Haifeng Wang**
Baidu Inc., Beijing, P. R. China
SurnameForename@baidu.com
[†]wu_hua@baidu.com

Abstract

This paper proposes a Markov Decision Process and reinforcement learning based approach for domain selection in a multi-domain Spoken Dialogue System built on a distributed architecture. In the proposed framework, the domain selection problem is treated as sequential planning instead of classification, such that confirmation and clarification interaction mechanisms are supported. In addition, it is shown that by using a model parameter tying trick, the extensibility of the system can be preserved, where dialogue components in new domains can be easily plugged in, without re-training the domain selection policy. The experimental results based on human subjects suggest that the proposed model marginally outperforms a non-trivial baseline.

1 Introduction

Due to growing demand for natural human-machine interaction, over the last decade Spoken Dialogue Systems (SDS) have been increasingly deployed in various commercial applications ranging from traditional call centre automation (e.g. AT&T "Lets Go!" bus information system (Williams et al., 2010)) to mobile personal assistants and knowledge navigators (e.g. Apple's Siri®, Google Now™, Microsoft Cortana, etc.) or voice interaction for smart household appliance control (e.g. Samsung Evolution Kit for Smart TVs). Furthermore, latest progress in open-vocabulary Automatic Speech Recognition (ASR) is pushing SDS from traditional single-domain information systems towards more complex multi-domain speech applications, of which typical examples are those voice assistant mobile applications.

Recent advances in SDS have shown that statistical approaches to dialogue management can result in marginal improvement in both the naturalness and the task success rate for domain-specific dialogues (Lemon and Pietquin, 2012; Young et al., 2013). State-of-the-art statistical SDS treat the dialogue problem as a sequential decision making process, and employ established planning models, such as Markov Decision Processes (MDPs) (Singh et al., 2002) or Partially Observable Markov Decision Processes (POMDPs) (Thomson and Young, 2010; Young et al., 2010; Williams and Young, 2007), in conjunction with reinforcement learning techniques (Jurčíček et al., 2011; Jurčíček et al., 2012; Gašić et al., 2013a) to seek optimal dialogue policies that maximise long-term expected (discounted) rewards and are robust to ASR errors.

However, to the best of our knowledge, most of the existing multi-domain SDS in public use are rule-based (e.g. (Gruber et al., 2012; Mirkovic and Cavedon, 2006)). The application of statistical models in multi-domain dialogue systems is still preliminary. Komatani et al. (2006) and Nakano et al. (2011) utilised a distributed architecture (Lin et al., 1999) to integrate expert dialogue systems in different domains into a unified framework, where a central controller trained as a data-driven classifier selects a domain expert at each turn to address user's query. Alternatively, Hakkani-Tür et al. (2012) adopted the well-known Information State mechanism (Traum and Larsson, 2003) to construct a multi-domain SDS and proposed a discriminative classification model for more accurate state updates. More recently, Gašić et al. (2013b) proposed that by a simple expansion of the kernel function in Gaussian Process (GP) reinforcement learning (Engel et al., 2005; Gašić et al., 2013a), one can adapt pre-trained dialogue policies to handle unseen slots for SDS in extended domains.

In this paper, we use a voice assistant applica-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 57–67,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

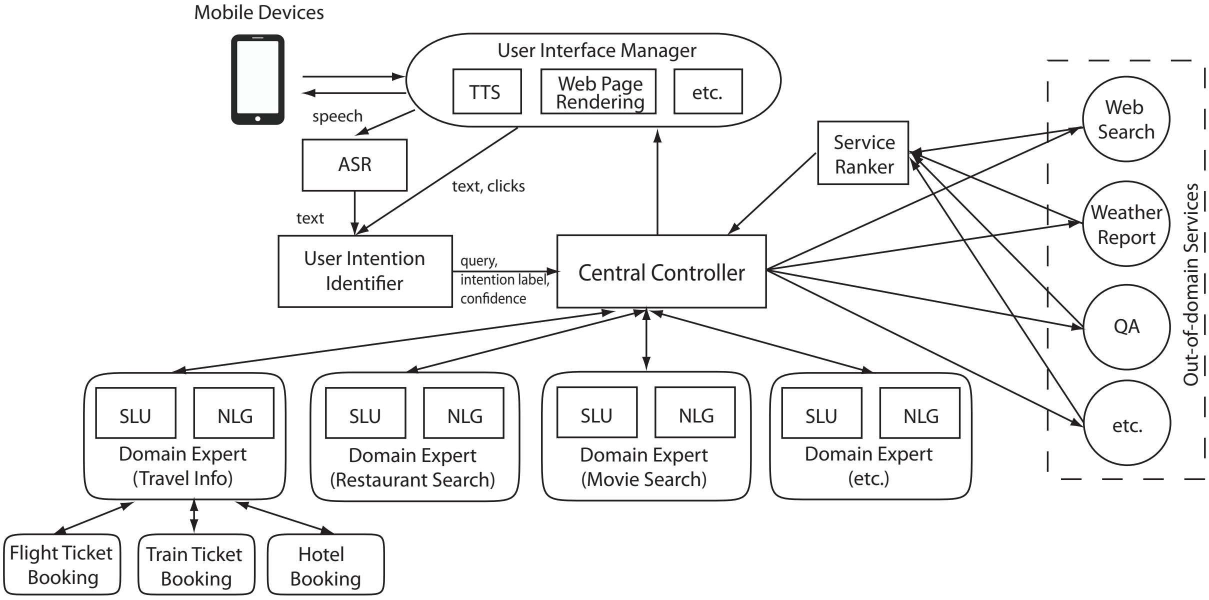

Figure 1: The distributed architecture of the voice assistant system (a simplified illustration).

tion (similar to Apple's Siri but in Chinese language) as an example to demonstrate a novel MDP-based approach for central interaction management in a complex multi-domain dialogue system. The voice assistant employs a distributed architecture similar to (Lin et al., 1999; Komatani et al., 2006; Nakano et al., 2011), and handles mixed interactions of multi-turn dialogues across different domains and single-turn queries powered by a collection of information access services (such as web search, Question Answering (QA), etc.). In our system, the dialogues in each domain are managed by an individual domain expert SDS, and the single-turn services are used to handle those so-called out-of-domain requests. We use featurised representations to summarise the current dialogue states in each domain (see Section 3 for more details), and let the central controller (the MDP model) choose one of the following system actions at each turn: (1) addressing user's query based on a domain expert, (2) treating it as an out-of-domain request, (3) asking user to confirm whether he/she wants to continue a domain expert's dialogue or to switch to out-of-domain services, and (4) clarifying user's intention between two domains. The Gaussian Process Temporal Difference (GPTD) algorithm (Engel et al., 2005; Gašić et al., 2013a) is adopted here for policy optimisation based on human subjects, where a parameter tying trick is applied to preserve the extensibility of the system, such that new domain experts (dialogue systems) can be flexibly plugged in without the need of re-training the central controller.

Comparing to the previous classification-based methods (Komatani et al., 2006; Nakano et al., 2011), the proposed approach not only has the advantage of action selection in consideration of long-term rewards, it can also yield more robust policies that allow clarifications and confirmations to mitigate ASR and Spoken Language Understanding (SLU) errors. Our human evaluation results show that the proposed system with a trained MDP policy achieves significantly better naturalness in domain switching tasks than a non-trivial baseline with a hand-crafted policy.

The remainder of this paper is organised as follows. Section 2 defines the terminology used throughout the paper. Section 3 briefly overviews the distributed architecture of our system. The MDP model and the policy optimisation algorithm are introduced in Section 4 and Section 5, respectively. After this, experimental settings and evaluation results are described in Section 6. Finally, we discuss some possible improvements in Section 7 and conclude ourselves in Section 8.

2 Terminology

A voice assistant application provides a unified speech interface to a collection of individual information access systems. It aims to collect and satisfy user requests in an interactive manner, where different types of interactions can be involved. Here we focus ourselves on two interaction scenarios, i.e. task-oriented (multi-turn) dialogues and single-turn queries.

According to user intentions, the dialogue interactions in our voice assistant system can further be categorised into different *domains*, of which each is handled by a separate dialogue manager, namely a *domain expert*. Example domains include travel information, restaurant search, etc. In addition, some domains in our system can be further decomposed into sub-domains, e.g. the travel information domain consists of three sub-domains: flight ticket booking, train ticket booking and hotel reservation. We use an integrated domain expert to address queries in all its sub-domains, so that relevant information can be shared across those sub-domains to allow intelligent induction in the dialogue flow.

For convenience of future reference, we call those single-turn information access systems *out-of-domain services* or simply *services* for short. The services integrated in our system include web search, semantic search, QA, system command execution, weather report, chat-bot, and many more.

3 System Architecture

The voice assistant system introduced in this paper is built on a distributed architecture (Lin et al., 1999), as shown in Figure 1, where the dialogue flow is processed as follows. Firstly, a user's query (either an ASR utterance or directly typed in text) is passed to a user intention identifier, which labels the raw query with a list of intention hypotheses with confidence scores. Here an intention label could be either a domain name or a service name. After this, the central controller distributes the raw query together with its intention labels and confidence scores to all the domain experts and the service modules, which will attempt to process the query and return their results to the central controller.

The domain experts in the current implementation of our system are all rule-based SDS following the RavenClaw framework proposed in (Bohus and Rudnicky, 2009). When receiving a query, a domain expert will use its own SLU module to parse the utterance or text input and try to update its dialogue state in consideration of both the SLU output and the intention labels. If the dialogue state in the domain expert can be updated given the query, it will return its output, internal session record and a confidence score to the central controller, where the output can be either a natural language utterance realised by its Natural Language Generation (NLG) module or a set of data records obtained from its database (if a database search operation is triggered), or both. If the domain expert cannot update its state using the current query, it will just return an empty result with a low confidence score. Similar procedures apply to those out-of-domain services as well, but there are no session records or confidence scores returned. Finally, given all the returned information, the central controller chooses, according to its policy, the module (either a domain expert or a service) whose results will be provided to the user.

When the central controller decides to pass a domain expert's output to the user, we regard the domain expert as being activated. Also note here, the updated state of a domain expert in a turn will not be physically stored, unless the domain expert is activated in that turn. This is a necessary mechanism to prevent an inactive domain expert being misled by ambiguous queries in other domains.

In addition, we use a well-engineered priority ranker to rank the services based on the numbers of results they returned as well as some prior knowledge about the quality of their data sources. When the central controller decides to show user the results from an out-of-domain service, it will choose the top one from the ranked list.

4 MDP Modelling of the Central Control Process

The main focus of this paper is to seek a policy for robustly switching the control flow among those domain experts and services (the service ranker in practice) during a dialogue, where the user may have multiple or compound goals (e.g. booking a flight ticket, booking a restaurant in the destination city and checking the weather report of the departure or destination city).

In order to make the system robust to ASR errors or ambiguous queries, the central controller should also have basic dialogue abilities for confirmation and clarification purposes. Here we define the confirmation as an action of asking whether a user wants to continue the dialogue in a certain domain. If the system receives a negative response at this point, it will switch to out-of-domain services. On the other hand, the clarification action is defined between domains, in which case, the system will explicitly ask the user to choose between two domain candidates before continuing the dialogue.

Due to the confirmation and clarification mechanisms defined above, the central controller becomes a sequential decision maker that must take the overall smoothness of the dialogue into account. Therefore, we propose an MDP-based approach for learning an optimal central control policy in this section.

The potential state space of our MDP is huge, which in principle consists of the combinations of all possible situations of the domain experts and the out-of-domain services, therefore function approximation techniques must be employed to enable tractable computations. However, when developing such a complex application as the voice assistant here, one also needs to take the extensibility of the system into account, so that new domain experts can be easily integrated into the system without major re-training or re-engineering of the existing components. Essentially, it requires the state featurisation and the central control policy learnt here to be independent of the number of domain experts. In Section 4.3, we show that such a property can be achieved by a parameter tying trick in the definition of the MDP.

4.1 MDP Preliminaries

Let $\mathcal{P}_X$ denote the set of probability distributions over a set X. An MDP is defined as a five tuple $\langle S, A, T, R, \gamma \rangle$, where the components are defined as follows. S and A are the sets of system states and actions, respectively. $T : S \times A \rightarrow \mathcal{P}_S$ is the transition function, and $T(s'|s, a)$ defines the conditional probability of the system transiting from state $s \in S$ to state $s' \in S$ after taking action $a \in A$. $R : S \times A \rightarrow \mathcal{P}_{\mathbb{R}}$ is the reward function with $R(s, a)$ specifying the distribution of the immediate rewards for the system taking action a at state s. In addition, $0 \leq \gamma \leq 1$ is the discount factor on the summed sequence of rewards.

A finite-horizon MDP operates as follows. The system occupies a state s and takes an action a, which then will make it transit to a next state $s' \sim T(\cdot|s, a)$ and receive a reward $r \sim R(s, a)$. This process repeats until a terminal state is reached.

For a given policy $\pi : S \rightarrow A$, the value function V^π is defined to be the expected cumulative reward, as $V^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{n} \gamma^t r_t|_{s_t,\pi(s_t)}\right]$, where s_0 is the starting state and n is the planning horizon. The aim of policy optimisation is to seek an optimal policy π^* that maximises the value function. If T and R are given, in conjunction with a Q-function, the optimal value V^* can be expressed by recursive equations as $Q(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s'|s, a)V^*(s')$ and $V^*(s) = \max_{a \in A} Q(s, a)$ (here we assume $R(s, a)$ is deterministic), which can be solved by dynamic programming (Bellman, 1957). For problems with unknown T or R, such as dialogue systems, the Q-values are usually estimated via reinforcement learning (Sutton and Barto, 1998).

4.2 Problem Definition

Let D denote the set of the domain experts in our voice assistant system, and s_d be the current dialogue state of domain expert $d \in D$ at a certain timestamp. We also define s_o as an abstract state to describe the current status of those out-of-domain services. Then mathematically we can represent the central control process as an MDP, where its state s is a joint set of the states of all the domain experts and the services, as $\mathrm{s} = \{s_d\}_{d \in D} \cup \{s_o\}$. Four types of system actions are defined as follows.

- `present`(d): presenting the output of domain expert d to user;
- `present_ood(null)`: presenting the results of the top-ranked out-of-domain service given by the service ranker;
- `confirm`(d): confirming whether user wants to continue with domain expert d (or to switch to out-of-domain services);
- `clarify`(d, d'): asking user to clarify his/her intention between domains d and d'.

For convenience of notations, we use $a(x)$ to denote a system action of our MDP, where $a \in \{\texttt{present}, \texttt{present\_ood}, \texttt{confirm}, \texttt{clarify}\}$, $x \in \{d, \texttt{null}, (d, d')\}_{d,d' \in D, d \neq d'}$, $x = \texttt{null}$ only applies to `present_ood`, and $x = (d, d')$ only applies to `clarify` actions.

4.3 Function Approximation

Function approximation is a commonly used technique to estimate the Q-values when the state space of the MDP is huge. Concretely, in our case, we assume that:

$$Q(\mathrm{s}, a(x)) = f(\phi(\mathrm{s}, a(x)); \theta) \qquad (1)$$

where $\phi : S \times A \rightarrow \mathbb{R}^K$ is a feature function that maps a state-action pair to an K-dimensional feature vector, and $f : \mathbb{R}^K \rightarrow \mathbb{R}$ is a function of $\phi(\mathbf{s}, a(x))$ parameterised by θ. A frequent choice of f is the linear function, as:

$$Q(\mathbf{s}, a(x)) = \theta^\top \phi(\mathbf{s}, a(x)) \tag{2}$$

After this, the policy optimisation problem becomes learning the parameter θ to approximate the Q-values based on example dialogue trajectories.

However, a crucial problem with the standard formulation in Eq. (2) is that the feature function ϕ is defined over the entire state and action spaces. In this case, when a new domain expert is integrated into the system, both the state space and the action space will be changed, therefore one will have to re-define the feature function and consequentially re-train the model. In order to achieve an extensible system, we make some simplification assumptions and decompose the feature function as follows. Firstly, we let:

$$\begin{aligned} \phi(\mathbf{s}, a(x)) &= \phi_a(\mathbf{s}_x) \qquad (3) \\ &= \begin{cases} \phi_{\texttt{pr}}(s_d) & \text{if } a(x) = \texttt{present}(d) \\ \phi_{\texttt{ood}}(s_o) & \text{if } a(x) = \texttt{present\_ood()} \\ \phi_{\texttt{cf}}(s_d) & \text{if } a(x) = \texttt{confirm}(d) \\ \phi_{\texttt{cl}}(s_d, s_{d'}) & \text{if } a(x) = \texttt{clarify}(d, d') \end{cases} \end{aligned}$$

where the feature function is reduced to only depend on the state of the action's operand, instead of the entire system state. Then, we make those actions $a(x)$ that have a same action type (a) but operate different domain experts (x) share the same parameter, i.e.:

$$Q(\mathbf{s}, a(x)) = \theta_a^\top \phi_a(\mathbf{s}_x) \tag{4}$$

This decomposition and parameter tying trick preserves the extensibility of the system, because both $\theta_a^\top$ and ϕ_a are independent of x, when there is a new domain expert $\tilde{d}$, we can directly substitute its state $s_{\tilde{d}}$ into Eq. (3) and (4) to compute its corresponding Q-values.

4.4 Features

Based on the problem formulation in Eq. (3) and (4), we shall only select high-level summary features to sketch the dialogue state and dialogue history of each domain expert, which must be applicable to all domain experts, regardless of their domain-specific characteristics or implementation differences. Suppose that the dialogue states of the domain experts can be represented as slot-value pairs[1], and for each domain there are required slots and optional slots, where all required slots must be filled before the domain expert can execute a database search operation. The features investigated in the proposed framework are listed in Table 1.

#	Feature	Range
1	the number of unfilled required slots of a domain expert	$\{0, \ldots, M\}$
2	the number of filled required slots of a domain expert	$\{0, \ldots, M\}$
3	the number of filled optional slots of a domain expert	$\{0, \ldots, L\}$
4	whether a domain expert has executed a database search	$\{0, 1\}$
5	the confidence score returned by a domain expert	$[0, 1.2]$
6	the total number of turns that a domain expert has been activated during a dialogue	$\mathbb{Z}^+$
7	e^{-t_a} where t_a denotes the relative turn of a domain expert being last activated, or 0 if not applicable	$[0, 1]$
8	e^{-t_c} where t_c denotes the relative turn of a domain expert being last confirmed, or 0 if not applicable	$[0, 1]$
9	the summed confidence score from the user intention identifier of a query being for out-of-domain services	$[0, 1.2N]$

Table 1: A list of all features used in our model. M and L respectively denote the maximum numbers of required and optional slots for the domain experts. N is the maximum number of hypotheses that the intention identifier can return. $\mathbb{Z}^+$ stands for the non-negative integer set.

Detailed featurisation in Eq. (3) is explained as follows. For $\phi_{\texttt{pr}}$, we choose the first 8 features plus a bias dimension that is always set to

---

[1] This is a rather general assumption. Informally speaking, for most task-oriented SDS, one can extract a slot-value representation from their dialogue models, of which examples include the RavenClaw architecture (Bohus and Rudnicky, 2009), the Information State dialogue engine (Traum and Larsson, 2003), MDP-SDS (Singh et al., 2002) or POMDP-SDS (Thomson and Young, 2010; Young et al., 2010; Williams and Young, 2007).

-1. Whilst, feature #9 plus a bias is used to define $\phi_{\texttt{ood}}$. All the features are used in $\phi_{\texttt{cf}}$, as to do a confirmation, one needs to consider the joint situation in and out of the domain. Finally, the feature function for a clarification action between two domains d and d' is defined as $\phi_{\texttt{cl}}(s_d, s_{d'}) = \exp\{-|\phi_{\texttt{pr}}(s_d) - \phi_{\texttt{pr}}(s_{d'})|\}$, where we use $|\cdot|$ to denote the element-wise absolute of a vector operand. The intuition here is that the more distinguishable the (featurised) states of two domain experts are, the less we tend to clarify them.

For those domain experts that have multiple sub-domains with different numbers of required and optional slots, the feature extraction procedure only applies to the latest active sub-domain.

In addition, note that, the confidence scores provided by the user intention identifier are only used as features for out-of-domain services. This is because in the current version of our system, the confidence estimation of the intention identifier for domain-dependent dialogue queries is less reliable due to the lack of context information. In contrast, the confidence scores returned by the domain experts will be more informative at this point.

5 Policy Learning with GPTD

In traditional statistical SDS, dialogue policies are usually trained using reinforcement learning based on simulated dialogue trajectories (Schatzmann et al., 2007; Keizer et al., 2010; Thomson and Young, 2010; Young et al., 2010). Although the evaluation of the simulators themselves could be an arguable issue, there are various advantages, e.g. hundreds of thousands of data examples can be easily generated for training and initial policy evaluation purposes, and different reinforcement learning models can be compared without incurring notable extra costs.

However, for more complex multi-domain SDS, particularly a voice assistant application like ours that aims at handling very complicated (ideally open-domain) dialogue scenarios, it would be difficult to develop a proper simulator that can reasonably mimic real human behaviours. Therefore, in this work, we learn the central control policy directly with human subjects, for which the following properties of the learning algorithm are required. Firstly and most importantly, the learner must be sample-efficient as the data collection procedure is costly. Secondly, the algorithm should support batch reinforcement learning. This is because when using function approximation, the learning process may not strictly converge, and the quality of the sequence of generated policies tends to oscillate after a certain number of improving steps at the beginning (Bertsekas and Tsitsiklis, 1996). If online reinforcement learning is used, we will be unable to evaluate the generated policy after each update, and hence will not know which policy to keep for the final evaluation. Therefore, we do a batch policy update and iterate the learning process for a number of batches, such that the data collection phase in a new iteration yields an evaluation of the policy obtained from the previous iteration at the same time.

To fulfill the above two requirements, the Gaussian Process Temporal Difference (GPTD) algorithm (Engel et al., 2005) is a proper choice, due to its sample efficiency (Fard et al., 2011) and batch learning ability (Engel et al., 2005), as well as its previous success in dialogue policy learning with human subjects (Gašić et al., 2013a). Note that, GPTD can also admit recursive (online) computations, but here we focus ourselves on the batch version.

A Gaussian Process (GP) is a generative model of Bayesian inference that can be used for function regression, and has the superiority of obtaining good posterior estimates with just a few observations (Rasmussen and Williams, 2006). GPTD models the Q-function as a zero mean GP which defines correlations in different parts of the featurised state and action spaces through a kernel function κ, as:

$$Q(\mathbf{s}, a(x)) \sim \mathcal{GP}(0, \kappa((\mathbf{s}_x, a), (\mathbf{s}_x, a))) \tag{5}$$

Given a sequence of t state-action pairs $\mathbf{X}_t = [(\mathbf{s}^0, a^0(x^0)), \ldots, (\mathbf{s}^t, a^t(x^t))]$ from a collection of dialogues and their corresponding immediate rewards $\mathbf{r}_t = [r^0, \ldots, r^t]$, the posterior of $Q(\mathbf{s}, a(x))$ for an arbitrary new state-action pair $(\mathbf{s}, a(x))$ can be computed as:

$$\begin{aligned} & Q(\mathbf{s}, a(x))|_{\mathbf{X}_t, \mathbf{r}_t} \\ & \quad \sim \ \mathcal{N}\left(\bar{Q}(\mathbf{s}, a(x)), \operatorname{cov}(\mathbf{s}, a(x))\right) && (6) \\ & \bar{Q}(\mathbf{s}, a(x)) = \mathbf{k}_t(\mathbf{s}_x, a)^\top \mathbf{H}_t^\top \mathbf{G}_t^{-1} \mathbf{r}_t && (7) \\ & \operatorname{cov}(\mathbf{s}, a(x)) = \kappa((\mathbf{s}_x, a), (\mathbf{s}_x, a)) \\ & \quad - \ \mathbf{k}_t(\mathbf{s}_x, a)^\top \mathbf{H}_t^\top \mathbf{G}_t^{-1} \mathbf{H}_t \mathbf{k}_t(\mathbf{s}_x, a) && (8) \\ & \mathbf{G}_t = \mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \sigma^2 \mathbf{H}_t \mathbf{H}_t^\top && (9) \end{aligned}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & -\gamma \end{bmatrix} \tag{10}$$

where $\mathbf{K}_t$ is the Gram matrix with elements $\mathbf{K}_t(i,j) = \kappa((\mathbf{s}^i_{x^i}, a^i), (\mathbf{s}^j_{x^j}, a^j))$, $\mathbf{k}_t(\mathbf{s}_x, a) = [\kappa((\mathbf{s}^i_{x^i}, a^i), (\mathbf{s}_x, a))]^t_{i=0}$ is a vector, and σ is a hyperparameter specifying the diagonal covariance values of the zero-mean Gaussian noise. In addition, we use $\mathrm{cov}\,(\mathbf{s}, a(x))$ to denote (for short) the self-covariance $\mathrm{cov}\,(\mathbf{s}, a(x), \mathbf{s}, a(x))$.

In our case, as different feature functions ϕ_a are defined for different action types, the kernel function is defined to be:

$$\kappa((\mathbf{s}_x, a), (\mathbf{s}'_{x'}, a')) = [\![a = a']\!]\kappa_a(\mathbf{s}_x, \mathbf{s}'_{x'}) \tag{11}$$

where $[\![\cdot]\!]$ is an indicator function and κ_a is the kernel function defined corresponding to the feature function ϕ_a.

Given a state, a most straightforward policy is to select the action that corresponds to the maximum mean Q-value estimated by the GP. However, since the objective is to learn the Q-function associated with the optimal policy by interacting directly with users, the policy must exhibit some form of stochastic behaviour in order to explore alternatives during the process of learning. In this work, the strategy employed for the exploration-exploitation trade-off is that, during exploration, actions are chosen according to the variance of the GP estimate for the Q-function, and during exploitation, actions are chosen according to the mean. That is:

$$\pi(\mathbf{s}) = \begin{cases} \arg\max_{a(x)} \bar{Q}(\mathbf{s}, a(x)) : \text{w.p. } 1 - \epsilon \\ \arg\max_{a(x)} \mathrm{cov}\,(\mathbf{s}, a(x)) \ : \text{w.p. } \epsilon \end{cases} \tag{12}$$

where $0 < \epsilon < 1$ is a pre-defined exploration rate, and will be exponentially reduced at each batch iteration during our learning process.

Note that, in practice, not all the actions are valid at every possible state. For example, if a domain expert d has never been activated during a dialogue and can neither process the user's current query, the actions with an operand d will be regarded as invalid at this state. When executing the policy, we only consider those valid actions for a given state.

Score	Interpretation
5	The domain selections are totally correct, and the entire dialogue flow is fluent.
4	The domain selections are totally correct, but the dialogue flow is slightly redundant.
3	There are accidental domain selections errors, or the dialogue flow is perceptually redundant.
2	There are frequent domain selections errors, or the dialogue flow is intolerably redundant.
1	Most domain selections are incorrect, or the dialogue is incompletable.

Table 2: The scoring standard in our experiments.

6 Experimental Results

6.1 Training

We use the batch version of GPTD as described in Section 5 to learn the central control policy with human subjects. There are three domain experts available in our current system, but during the training only two domains are used, which are the travel information domain and the restaurant search domain. We reserve a movie search domain for evaluating the generalisation property of the learnt policy (see Section 6.2). The learning process started from a hand-crafted policy. Then 15 experienced users[2] volunteered to contribute dialogue examples with multiple or compound goals (see Figure 4 for an instance), from whom we collected around 50∼70 dialogues per day for 5 days[3]. After each dialogue, the users were asked to score the system from 5 to 1 according to a scoring standard shown in Table 2. The scores were taken as the (delayed) rewards to train the GPTD model, where we set the rewards for intermediate turns to 0. The working policy was updated daily based on the data obtained up to that day. The data collected on the first day was used for preliminary experiments to choose the hyperparame-

---

[2] Overall user satisfactions may rely on various aspects of the entire system, e.g. the data source quality of the services, the performance of each domain expert, etc. It will be difficult to make non-experienced users to score the central controller isolatedly.

[3] Not all the users participated the experiments everyday. There were 311 valid dialogues received in total, with an average length of 9 turns.

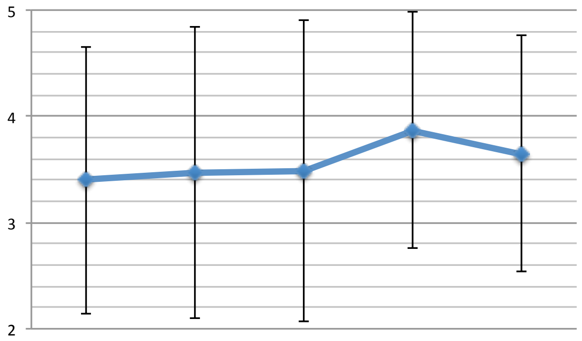

Figure 2: Average scores and standard deviations during policy iteration.

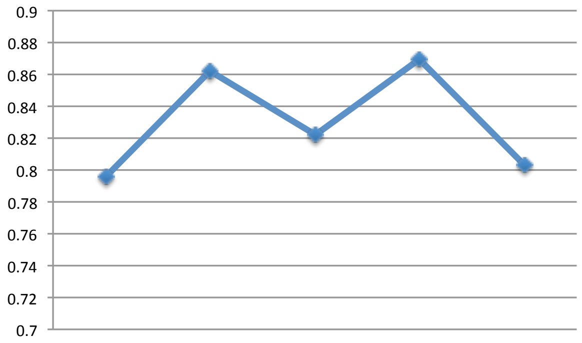

Figure 3: Domain selection accuracies during policy iteration.

ters of the model, such as the kernel function, the kernel parameters (if applicable), and σ and γ in the GPTD model. We initially experimented with linear, polynomial and Gaussian kernels, with different configurations of σ and γ values, as well as kernel parameter values. It was found that the linear kernel in combination with $\sigma = 5$ and $\gamma = 0.99$ works more appropriate than the other settings. This configuration was then fixed for the rest iterations.

The learning process was iterated for 4 days after the first one. On each day, we computed the mean and standard deviation of the user scores as an evaluation of the policy learnt on the previous day. The learning curve is illustrated in Figure 2. Note here, as we were actually executing a stochastic policy according to Eq. (12), to calculate the values in Figure 2 we ignored those dialogues that contain any actions selected due to the exploration. We also manually labelled the correctness of domain selection at every turn of the dialogues. The domain selection accuracies of the obtained policy sequence are shown in Figure 3, where similarly, those exploration actions as well as the clarification and confirmation actions were excluded from the calculations. Although the domain selection accuracy is not the target that our learning algorithm aims to optimise, it reflects the quality of the learnt policies from a different angle of view.

Scenario	Policy		p-value
	Baseline	GPTD	
(i)	4.5±0.8	4.2±0.8	0.387
(ii)	3.4±0.9	4.2±0.8	0.018
(iii)	4.1±1.0	4.3±1.0	0.0821
(iv)	3.9±1.1	4.5±0.8	0.0440

Table 3: Paired comparison experiments between the system with a trained GPTD policy and the rule-based baseline.

It can be found in Figure 2 that the best policy was obtained in the third iteration, and after that the policy quality oscillated. The same finding is indicated in Figure 3 as well, when we use the domain selection accuracy as the evaluation metric. Therefore, we kept the policy corresponding to the peak point of the learning curve for the comparison experiments below.

6.2 Comparison Experiments

We conducted paired comparison experiments in four scenarios to compare between the system with the GPTD-learnt central control policy and a non-trivial baseline. The baseline is a publicly deployed version of the voice assistant application. The central control policy of the baseline system is handcrafted, which has a separate list of semantic matching rules for each domain to enable domain switching.

The first two scenarios aim to test the performance of the two systems on (i) switching between a domain expert and out-of-domain services, and (ii) switching between two domain experts, where only the two training domains (travel information and restaurant search) were considered. Scenarios (iii) and (iv) are similar to scenarios (i) and (ii) respectively, but at this time, the users were required to carry out the tests surrounding the movie search domain (which is addressed by a new domain expert not used in the training phase). There were 13 users who participated this experiment. In each scenario, every user was required to test the two systems with an identical goal and similar queries. After each test, the users were asked to score the two systems separately according to the scoring standard in Table 2.

The average scores received by the two systems are shown in Table 3, where we also compute the statistical significance (the p-values) of the results based on paired t-tests. It can be found that the learnt policy works significantly better than the rule-based policy in scenarios (ii) and (iv), but in scenarios (i) and (iii) the differences between two systems are statistically insignificant. Moreover, the learnt policy preserves the extensibility of the entire system as expected, of which strong evidences are given by the results in scenarios (iii) and (iv).

6.3 Policy Analysis

To better understand the policy learnt by the GPTD model, we look into the obtained weight vectors, as shown in Table 4. It can be found that confidence score (#5) is an informative feature for all the system actions, while the relative turn of a domain being last activated (#7) is an additional strong evidence for a confirmation decision. In addition, the similarity between the dialogue completion status (#1 & #2) of two ambiguous domain experts and the relative turns of them being last confirmed (#8) tend to be extra dominating features for clarification decisions, besides the closeness of the confidence scores returned by the two domain experts.

A less noticeable but important phenomenon is observed for feature #6, i.e. the total number of active turns of a domain expert during a dialogue. Concretely, feature #6 has a small negative effect on presentation actions but a small positive contribution to confirmation actions. Such weights could correspond to the discount factor's penalty to long dialogues in the value function. However, it implies an unexpected effect in extreme cases, which we explain in detail as follows. Although the absolute weights for feature #6 are tiny for both presentation and confirmation actions, the feature value will grow linearly during a dialogue. Therefore, when a dialogue in a certain domain last rather long, there tend to be very frequent confirmations. A possible solution to this problem could be either ignoring feature #6 or twisting it to some nonlinear function, such that its value stops increasing at a certain threshold point. In addition, to cover sufficient amount of those "extreme" examples in the training phase could also be an alternative solution, as our current training set contains very few examples that exhibit extraordinary long domain persistence.

#	Feature Weights			
	present	confirm	clarify	
1	0.09	0.02	0.60	present_ood
2	0.20	0.29	0.53	
3	0.18	0.29	0.16	
4	-0.10	0.16	0.25	
5	0.75	0.57	0.54	
6	-0.02	0.11	0.13	
7	0.25	1.19	0.36	
8	-0.22	-0.19	0.69	
9	–	0.20	–	0.47
Bias	-1.79	–	–	-2.42

Table 4: Feature weights learnt by GPTD. See Table 1 for the meanings of the features.

7 Further Discussions

The proposed approach is a rather general framework to learn extensible central control policies for multi-domain SDS based on distributed architectures. It does not rely on any internal representations of those individual domain experts, as long as a unified featurisation of their dialogue states can be achieved.

However, from the entire system point of view, the current implementation is still preliminary. Particularly, the confirmation and clarification mechanisms are isolated, for which the surface realisations sometimes may sound stiff. This phenomenon explains one of the reasons that make the proposed system slightly less preferred by the users than the baseline in scenario (i), when the interaction flows are relative simple. A possible improvement here could be associating the confirmation and clarification actions in the central controller to the error handling mechanisms within each domain expert, and letting domain experts generate their own utterances on receiving a confirmation/clarification request from the central controller.

Online reinforcement learning with real user cases will be another undoubted direction of further improvement of our system. The key challenge here is to automatically estimate user's satisfactions, which will be transformed to the rewards for the reinforcement learners. Strong feedbacks such as clicks or actual order placements can be collected. But to regress user’s true satisfaction, other environment features must also be taken into account. Practical solutions are still an open issue at this stage, and are left to our future work.

8 Conclusion

In this paper, we introduce an MDP framework for learning domain selection policies in a complex multi-domain SDS. Standard problem formulation is modified with tied model parameters, so that the entire system is extensible and new domain experts can be easily integrated without re-training the policy. This expectation is confirmed by empirical experiments with human subjects, where the proposed system marginally beats a non-trivial baseline and demonstrates proper extensibility. Several possible improvements are discussed, which will be the central arc of our future research.

Acknowledgements

The research in this paper is supported by China’s 973 Programme no. 2014CB340505. The first author is partially funded by the EC FP7 programme under grant agreement no. 287615 (PARLANCE) and a SICSA PECE grant. The authors would also like to thank Qiaoqiao She, Duo Cai and the HCI-APP group at Baidu for volunteering to participate in the human subject experiments.

USER/ASR: 去青岛的机票 (Flight ticket to Qingdao.)

SYSTEM: 好的，到青岛的飞机票。请问您什么时候离开？
自哪儿出发啊？
(Okay, flight ticket to Qingdao. When are you
departing, and where are you departing from?)
[Travel Info. Domain Expert]

USER/ASR: 明天出发 (I’m leaving tomorrow.)

SYSTEM: 好的，明天到青岛的飞机票。您预计由哪儿出发啊？
(Okay, flight ticket to Qingdao, leaving
tomorrow. Where are you departing from?)
[Travel Info. Domain Expert]

USER/ASR: 北京 (Beijing)

SYSTEM: [Results Displayed ☰ - Travel Info. Domain Expert]

USER: 青岛有好吃的四川菜吗
(Are there any good Sichuan cuisines in Qingdao?)
ASR: 听到有好吃的酸的川菜吗
(ungrammatical utterance, partially with
“delicious” and “Sichuan food”.)

SYSTEM: 您要继续订机票吗？
(Do you want to continue with flight ticket booking?)
[Confirmation - Central Controller]

USER: 不是 (No)

SYSTEM: [Web Search Service: “听到有好吃的酸的川菜吗” 🔍]

USER/ASR: 明天去青岛 (I’m going to Qingdao tomorrow.)

SYSTEM: [Results Displayed ☰ - Travel Info. Domain Expert]

USER/ASR: 青岛机场附近的餐厅 (Restaurants near Qingdao Airport)

SYSTEM: 为您找到以下餐厅
(I found the following restaurants for you.)
[Results Displayed 🗺 - Restaurant Search Domain Expert]

USER/ASR: 青岛明天会下雨吗? (Will it rain in Qingdao tomorrow?)

SYSTEM: 青岛明天阵雨，温度16℃至20℃，东风5-6级
(Qingdao is rainy tomorrow. The temperature is 16 to 20
degree centigrade with moderate east wind.)
[Results Displayed 🌧 - Weather Report Service]

Figure 4: An example dialogue containing multiple user goals. The icons stand for graphical user interface based nonverbal outputs.

References

Richard Ernest Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Dimitri P. Bertsekas and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Dan Bohus and Alexander I. Rudnicky. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23(3):332–361.

Yaakov Engel, Shie Mannor, and Ron Meir. 2005. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 201–208.

Mahdi Milani Fard, Joelle Pineau, and Csaba Szepesvári. 2011. PAC-Bayesian policy evaluation for reinforcement learning. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 195–202.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013a. Online policy optimisation of Bayesian spoken dialogue systems via human interaction. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8367–8371.

Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013b. POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the 14th annual SIGdial Meeting on Discourse and Dialogue*, pages 214–222.

Thomas Robert Gruber, Adam John Cheyer, Dag

Kittlaus, Didier Rene Guzzoni, Christopher Dean Brigham, Richard Donald Giuli, Marcello Bastea-Forte, and Harry Joseph Saddler. 2012. Intelligent automated assistant. United States Patent No. US 20120245944 A1.

Dilek Z. Hakkani-Tür, Gokhan Tür, Larry P. Heck, Ashley Fidler, and Asli Çelikyilmaz. 2012. A discriminative classification-based approach to information state updates for a multi-domain dialog system. In *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2011. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs. *ACM Transactions on Speech and Language Processing*, 7(3):6:1–6:25.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2012. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 26(3):168–192.

Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Parameter estimation for agenda-based user simulation. In *Proceedings of the 11th annual SIGdial Meeting on Discourse and Dialogue*, pages 116–123.

Kazunori Komatani, Naoyuki Kanda, Mikio Nakano, Kazuhiro Nakadai, Hiroshi Tsujino, Tetsuya Ogata, and Hiroshi G. Okuno. 2006. Multi-domain spoken dialogue system with extensibility and robustness against speech recognition errors. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 9–17.

Oliver Lemon and Olivier Pietquin, editors. 2012. *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces*. Springer.

Bor-shen Lin, Hsin-min Wang, and Lin-Shan Lee. 1999. A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Danilo Mirkovic and Lawrence Cavedon. 2006. Dialogue management using scripts. United States Patent No. US 20060271351 A1.

Mikio Nakano, Shun Sato, Kazunori Komatani, Kyoko Matsuyama, Kotaro Funakoshi, and Hiroshi G. Okuno. 2011. A two-stage domain selection framework for extensible multi-domain spoken dialogue systems. In *Proceedings of the 12th annual SIGdial Meeting on Discourse and Dialogue*, pages 18–29.

Carl Edward Rasmussen and Christopher K. I. Williams, editors. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16(1):105–133.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588.

David R. Traum and Staffan Larsson. 2003. The Information State approach to dialogue management. In Jan van Kuppevelt and Ronnie W. Smith, editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Springer.

Jason D. Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Jason D. Williams, Iker Arizmendi, and Alistair Conkie. 2010. Demonstration of AT&T “Let’s Go”: A production-grade statistical spoken dialog system. In *Proceedings of the 3rd IEEE Workshop on Spoken Language Technology (SLT)*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialogue systems: a review. *Proceedings of the IEEE*, PP(99):1–20.