

Learning Domain-Independent Dialogue Policies via Ontology Parameterisation

Zhuoran Wang¹, Yannis Stylianou¹, Tsung-Hsien Wen², Pei-Hao Su², Steve Young²

¹Toshiba Research Europe Ltd., Cambridge, UK

²Engineering Department, University of Cambridge, UK

Abstract

This paper introduces a novel approach to eliminate the domain dependence of dialogue state and action representations, such that dialogue policies trained based on proposed representations can be transferred across different domains. The experimental results show that the policy optimised in a restaurant search domain using our domain-independent representations can be deployed to a laptop sale domain, achieving a performance very close to that of the policy optimised directly using in-domain dialogues.

1 Introduction

Statistical approaches to Spoken Dialogue Systems (SDS), particularly Partially Observable Markov Decision Processes (POMDPs) (Lemon and Pietquin, 2012; Young et al., 2013), have demonstrated great success in improving the robustness of dialogue policies to error-prone Automatic Speech Recognition (ASR). However, building statistical SDS (SSDS) for different application domains is time consuming. Traditionally, each component of such SSDS needs to be trained based on domain-specific data, which are not always easy to obtain. Moreover, in many cases, one will need a basic (e.g. rule-based) working SDS to be built before starting the data collection procedure, whereas developing the initial system for a new domain requires a significant amount of human expertise.

It will be undoubtedly beneficial to SDS development if one could either learn domain-independent knowledge representations or transfer the knowledge learnt previously in existing domains to new domains. Existing works in this respect include domain-independent intermediate semantic extractors for Spoken Language Understanding (SLU), domain-general formulae (Wang

and Lemon, 2013; Sun et al., 2014) and domain-extensible delexicalised deep classifiers (Henderson et al., 2014) for dialogue state tracking, and domain-extensible/transferrable Gaussian Process (GP) based dialogue policies (Gašić et al., 2013; Gašić et al., 2015), to name but a few.

In this paper, we introduce a more effective approach to eliminate domain dependence of dialogue policies, by exploring the nature and commonness of the underlying tasks of SDS in different domains and parameterising different slots defined in the domain ontologies into a common feature space according to their relations and potential contributions to the underlying tasks. For the ease of access to the proposed technique (§3), we start from a brief review of POMDP-SDS in §2. Promising experimental results are achieved based on both simulated users and human subjects as shown in §4, followed by further discussions and conclusions (§5).

2 POMDP-SDS: A Brief Overview

POMDP is a powerful tool for modelling sequential decision making problems under uncertainty, by optimising the policy to maximise long-term cumulative rewards. Concretely, at each turn of a dialogue, a typical POMDP-SDS parses an observed ASR n -best list with confidence scores into semantic representations (again with associated confidence scores), and estimates a distribution over (unobservable) user goals, called a belief state. After this, the dialogue policy selects a semantic-level system action, which will be realised by Natural Language Generation (NLG) before synthesising the speech response to the user.

The semantic representations in SDS normally consist of two parts, a communication function (e.g. *inform*, *deny*, *confirm*, etc.) and (optionally) a list of slot-value pairs (e.g. *food=pizza*, *area=centre*, etc.). The prior knowledge defining the slot-values in a particular

domain is called the domain ontology.

Dialogue policy optimisation can be solved via Reinforcement Learning (RL), where the aim is to estimate a quantity $Q(\mathbf{b}, \mathbf{a})$, for each \mathbf{b} and \mathbf{a} , reflecting the expected cumulative rewards of the system executing action \mathbf{a} at belief state \mathbf{b} , such that the optimal action \mathbf{a}^* can be determined for a given \mathbf{b} according to $\mathbf{a}^* = \arg \max_{\mathbf{a}} Q(\mathbf{b}, \mathbf{a})$. Due the exponentially large state-action space an SDS can incur, function approximation is necessary, where it is assumed that $Q(\mathbf{b}, \mathbf{a}) \approx f_{\theta}(\phi(\mathbf{b}, \mathbf{a}))$, where θ is the model parameter to learn, and $\phi(\cdot)$ is a feature function that maps (\mathbf{b}, \mathbf{a}) to a feature vector. To compute $Q(\mathbf{b}, \mathbf{a})$, one can either use a low-dimensional summary belief (Williams and Young, 2005) or the full belief itself if kernel methods are applied (Gašić et al., 2012). But in both cases, the action \mathbf{a} will be a summary action (see §3 for more details) to achieve tractable computations.

3 Domain-Independent Featurisation

For the convenience of further discussion, we firstly take a closer look at how summary actions can be derived from their corresponding master actions. Assume that according to its communication function, a system action \mathbf{a} can take one of the following forms: $a()$ (e.g. `reqmore()`), $a(s)$ (e.g. `request(food)`), $a(s = v)$ (e.g. `confirm(area=north)`), $a(s = v_1, s = v_2)$ (e.g. `select(food=noodle, food=pizza)`), and $a(s_1 = v_1, \dots, s_n = v_n)$ (e.g. `offer(name="Chop Chop", food=Chinese)`), where a stands for the communication function, s_* and v_* denote slots and values respectively. Usually it is unnecessary for the system to address a hypothesis less believable than the top hypothesis in the belief (or the top two hypotheses in the ‘select’ case). Therefore, by abstracting the actual values, the system actions can be represented as $a(s = \mathbf{b}_s^{\text{top}})$, $a(s = \mathbf{b}_s^{\text{top}}, s = \mathbf{b}_s^{\text{second}})$ and $a(\mathbf{b}_{\text{joint}}^{\text{top}})$, where \mathbf{b}_s denotes the marginal belief with respect to slot s , $\mathbf{b}_{\text{joint}}$ stands for the joint belief, and $\mathbf{b}_*^{\text{top}}$ and $\mathbf{b}_*^{\text{second}}$ denote the top and second hypotheses of a given \mathbf{b}_* , respectively. After this, summary actions can be defined as a_s (for actions depending on s) and a (for those having no operands or only taking joint hypotheses as operands, i.e. independent of any particular slot). Furthermore, one can uniquely map such summary actions back to their

master actions, by substituting the respective top (and second if necessary) hypotheses in the belief into the corresponding slots.

Based on the above definition, we can re-write the master action \mathbf{a} as \mathbf{a}_s , where s denotes the slot that \mathbf{a} depends on when summarised. Here, s is fully derived from \mathbf{a} and can be null (when the summary action of \mathbf{a} is just a). A conventional form of ϕ can be expressed as $\phi(\mathbf{b}, \mathbf{a}_s) = \delta(a_s) \otimes \psi(\mathbf{b})$ where δ is the Kronecker delta, \otimes is the tensor product, and generally speaking, $\psi(\cdot)$ featurises the belief state, which can yield the summary belief in particular cases.

3.1 “Focus-aware” belief summarisation

Without losing generality, one can assume that the communication functions a are domain-independent. However, since the slots s are domain-specific (defined by the ontology), both a_s and \mathbf{b} will be domain-dependent.

Making $\psi(\mathbf{b})$ domain-independent can be trivial. A commonly used representation of \mathbf{b} consists of a set of individual belief vectors, denoted as $\{\mathbf{b}_{\text{joint}}, \mathbf{b}_o\} \cup \{\mathbf{b}_s\}_{s \in S}$, where \mathbf{b}_o stands for the section of \mathbf{b} independent of any slots (e.g. the belief over communication methods, such as “by constraint”, “by name”, etc. (Thomson and Young, 2010)) and S stands for the set of (informable) slots defined in the domain ontology. One can construct a feature function $\psi(\mathbf{b}, s) = \psi_1(\mathbf{b}_{\text{joint}}) \oplus \psi_2(\mathbf{b}_o) \oplus \psi_3(\mathbf{b}_s)$ for a given s and let $\phi(\mathbf{b}, \mathbf{a}_s) = \delta(a_s) \otimes \psi(\mathbf{b}, s)$, where \oplus stands for the operator to concatenate two vectors. (In other words, the belief summarisation here only focuses on the slot being addressed by the proposed action, regardless of the beliefs for other slots.) As the mechanism in each ψ_* to featurise its operand \mathbf{b}_* can be domain-independent (see Section 3.3 for an example), the resulting overall feature vector will be domain-general.

3.2 Ontology (slot) parameterisation

If we could further parameterise each slot s in a domain-general way (as $\varphi(s)$), and define

$$\phi(\mathbf{b}, \mathbf{a}_s) = \delta(a) \otimes [\varphi_a(s) \oplus \psi_a(\mathbf{b}, s)] \quad (1)$$

the domain dependence of the overall feature function ϕ will be eliminated¹. Note here, to make the

¹An alternative featurisation can be $\phi(\mathbf{b}, \mathbf{a}_s) = \delta(a) \otimes \varphi_a(s) \otimes \psi_a(\mathbf{b}, s)$, but our preliminary experiments show that \otimes results in similar but slightly worse policies. Therefore, we stick on \oplus in this paper.

definition more general, we assume that the feature functions φ_a and ψ_a depend on a , such that a different featurisation can be applied for each a .

To achieve a meaningful parameterisation $\varphi_a(s)$, we need to investigate how each slot s is related to completing the underlying task. More concretely, for example, if the underlying task is to obtain user’s constraint on each slot so that the system can conduct a database (DB) search to find suitable entities (e.g. venues, products, etc.), then the slot features should describe the potentiality of the slot to refine the search results (reduce the number of matching entities) if that slot is filled. For another example, if the task is to gather necessary (plus optional) information to execute a system command (e.g. setting a reminder or planning a route), where the number of values of each slot can be unbounded, then the slots features should indicate whether the slot is required or optional. In addition, the slots may have some specific characteristics causing people addressing them differently in a dialogue. For example, when buying a laptop, more likely one would talk about the price first than the battery rating. Therefore, features describing the priority of each slot are also necessary to yield natural dialogues. We give a complete list of features in next subsection for a working example, to demonstrate how two irrelevant domains can share a common ontology parameterisation.

3.3 A Working Example

We use restaurant search and laptop sale as two example domains to explain the above idea. The underlying tasks of the both problems can be regarded as DB search. Appendix A gives the detailed ontology definitions of the two domains.

Firstly, the following notations are introduced for the convenience of discussion. Let V_s denote the set of the values that a slot s can take, and $|\cdot|$ be the size of a set. Assume that $h = (s_1 = v_1 \dots \wedge s_n = v_n)$ is a user goal hypothesis consisting a set of slot-value pairs. We use $\text{DB}(h)$ to denote the set of the entities in the DB satisfying h . In addition, we define $\lfloor x \rfloor$ to be the largest integer less than and equal to x . After this, for each informable slot s defined in Table ??, the following quantities are used for its parameterisation.

- **Number of values**

- a continuous feature², $1/|V_s|$;

²The normalised quantity is used here to make this fea-

- discrete features mapping $|V_s|$ into N bins, indexed by $\min\{\lfloor \log_2 |V_s| \rfloor, N\}$.

- **Importance:** two features describing, respectively, how likely a slot will and will not occur in a dialogue.

- **Priority:** three features denoting, respectively, how likely a slot will be the first, the second, and a later attribute to address in a dialogue.

- **Value distribution in the DB:** the entropy of the normalised histogram, $(|\text{DB}(s = v)|/|\text{DB}|)_{v \in V_s}$.

- **Potential contribution to DB search:** given the current top user goal hypothesis h^* and a pre-defined threshold τ

- how likely filling s will reduce the number of matching DB records to below τ , i.e. $|\{v : v \in V_s, |\text{DB}(h^* \wedge s = v)| \leq \tau\}|/|V_s|$;
- how likely filling s will not reduce the number of matching DB records to below τ , i.e. $|\{v : v \in V_s, |\text{DB}(h^* \wedge s = v)| > \tau\}|/|V_s|$;
- how likely filling s will result in no matching records found in the DB, i.e. $|\{v : v \in V_s, \text{DB}(h^* \wedge s = v) = \emptyset\}|/|V_s|$.

The importance and priority features used in this work are manually assigned binary values, but ideally, if one has some in-domain human dialogue examples (e.g. from Wizard-of-Oz experiments), such feature values can be derived from simple statistics on the corpus. In addition, we make the last set of features only applicable to those slots not observed in the top joint hypothesis.

The summary belief features used in this work are sketched as follows. For each informable slot s and each of its applicable action types a , $\psi_a(\mathbf{b}, s)$ extracts the probability of $\mathbf{b}_s^{\text{top}}$, the entropy of \mathbf{b}_s , the probability difference between the top two marginal hypotheses (discretised into 5 bins with interval size 0.2) and the non-zero rate ($|\{v : v \in V_s, \mathbf{b}_s(v) > 0\}|/|V_s|$). In addition, if the slot is requestable, the probability of it being requested by the user (Thomson and Young, 2010) is used as an extra feature. A similar featurisation procedure (except the “requested” probability) is applied to the joint belief as well, from which the obtained features are used for all communication functions. To capture the nature of the underlying task (DB

ture have a similar value range to the others, for numerical stability purposes in GP-based policy learning (see §4).

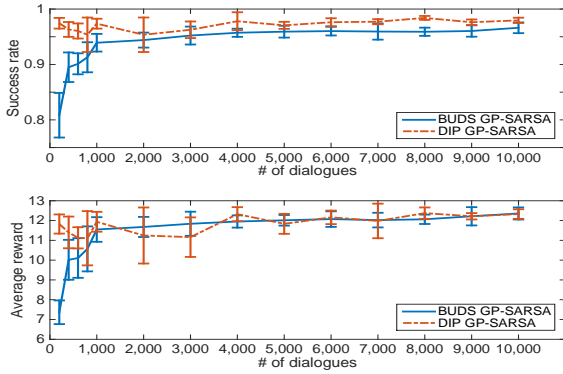


Figure 1: Training GP-SARSA policies for BUDS (full belief) and DIP in the restaurant search domain. Each point is averaged over 5 policies each evaluated on 1000 simulated dialogues.

search), we define two additional features for the joint belief, an indicator $\mathbb{I}[|DB(\mathbf{b}_{\text{joint}}^{\text{top}})| \leq \tau]$ and a real-valued feature $|DB(\mathbf{b}_{\text{joint}}^{\text{top}})|/\tau$ if the former is false, where τ is the same pre-defined threshold used for slot parameterisation as introduced above. There are also a number of slot-independent features applied to all action types, including the belief over communication methods (Thomson and Young, 2010) and the marginal confidence scores of user dialogue act types in the current turn.

4 Experimental Results

In the following experiments, we integrate the proposed domain-independent parameterisation (DIP) method with a generic dialogue state tracker (Wang and Lemon, 2013) to yield an overall domain-independent dialogue manager. Firstly, we train DIP dialogue policies in the restaurant search domain using GP-SARSA based on a state-of-the-art agenda-based user simulator³ (Schatzmann et al., 2007), in comparison with the GP-SARSA learning process for the well-known BUDS system (Thomson and Young, 2010) (where full beliefs are used (Gašić and Young, 2014)), as shown in Figure 1. It can be found that the proposed method results in faster convergence and can even achieve slightly better performance than the conventional approach.

After this, we directly deployed the DIP policies trained in the restaurant domain to the laptop search domain, and compare its performance

³For all the experiments in this work, the confusion rate of the simulator is set to 15% and linear kernels are used for GP-SARSA.

| System | Reward | Success (%) | #Turns |
|----------------------------|----------|-------------|---------|
| DIP _{in-domain} | 12.5±0.3 | 98.3±1.2 | 7.2±0.3 |
| DIP _{transferred} | 12.2±0.4 | 97.8±0.9 | 7.4±0.3 |

Table 1: Policy evaluations in the laptop sale domain based on simulated dialogues.

| System | #Dials | Success (%) | Score |
|----------------------------|--------|-------------|-------|
| DIP _{in-domain} | 118 | 87.3 | 4.53 |
| DIP _{transferred} | 132 | 86.4 | 4.92 |

Table 2: Policy evaluations using human subjects.

with an in-domain policy trained using the simulator (configured to the laptop search domain). Table 1 shows that the performance of the transferred policy is almost identical to the in-domain policy.

Finally, we choose the best in-domain and transferred DIP policies and deploy them into respective end-to-end laptop sale SDS, for human subject experiments based on MTurk. After each dialogue, the user was also asked to provide a subjective score for the naturalness of the interaction, ranging from 1 (very unnatural) to 6 (very natural). The results are summarised in Table 2, where the transferred policy has an objective task success rate only about 1% lower than the in-domain policy. Moreover, very interestingly, the crowdsourcing users on average regard the transferred policy as slightly more natural than the in-domain policy.

5 Discussion and Conclusion

This paper proposed a domain-independent ontology parameterisation framework to enable domain-transfer of optimised dialogue policies. When compared to a closely related method introduced in (Gašić et al., 2013; Gašić et al., 2015), the proposed DIP mechanism directly addresses the nature of the underlying tasks and provides a more flexible way to parametrically measure the similarity between different domain ontologies, instead of manually tying slots in different domains. Experimental results show that when transferred to a new domain, dialogue policies trained based on the DIP representations can achieve comparable performance to those policies optimised using in-domain dialogues. Bridging the (very small) performance gap here should also be simple, if one takes the transferred policy as the prior and conducts domain-adaptation similar to (Gašić et al., 2015). This will be addressed in our future work.

References

- Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(1):28–40.
- Milica Gašić, Matthew Henderson, Blaise Thomson, Pirros Tsiakoulis, and Steve J. Young. 2012. Policy optimisation of POMDP-based dialogue systems without state space compression. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 31–36.
- Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. POMDP-based dialogue manager adaptation to extended domains. In *Proceedings of the SIGDIAL 2013 Conference*, pages 214–222.
- Milica Gašić, Dongho Kim, Pirros Tsiakoulis, and Steve Young. 2015. Distributed dialogue policies for multi-domain statistical dialogue management. In *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Matthew Henderson, Blaise Thomson, and Steve J. Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 360–365.
- Oliver Lemon and Olivier Pietquin, editors. 2012. *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces*. Springer.
- 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.
- Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014. A generalized rule based tracker for dialogue state tracking. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 330–335.
- 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.
- Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the Dialog State Tracking Challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432.
- Jason D. Williams and Steve Young. 2005. Scaling up POMDPs for dialog management: The “Summary POMDP” method. In *ASRU*.

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.

A Ontology Definitions for the Example Domains

| | Slot | #Values | Info. | Req. |
|------------|---------------|---------|-------|------|
| Restaurant | food | 91 | yes | yes |
| | area | 5 | yes | yes |
| | pricerange | 3 | yes | yes |
| | name | 111 | yes | yes |
| | phone | – | no | yes |
| | postcode | – | no | yes |
| | signature | – | no | yes |
| | description | – | no | yes |
| Laptop | family | 5 | yes | no |
| | purpose | 2 | yes | yes |
| | pricerange | 3 | yes | no |
| | weighrange | 3 | yes | no |
| | batteryrating | 3 | yes | yes |
| | driverange | 3 | yes | no |
| | name | 123 | yes | no |
| | price | – | no | yes |
| | hard drive | – | no | yes |
| | dimension | – | no | yes |

Table A.1: Ontologies for the restaurant search and laptop sale domains. “Info.” denotes informable slots, for which user can provide values; “Req.” denotes requestable slots, for which user can ask information.