Multi-agent learning in multi-domain spoken dialogue systems

Cambridge University Engineering Department
Trumpington St, Cambridge CB2 1PZ, UK
{mg436,nm480,lmr46,phs26,djv27,thw28}@cam.ac.uk

Abstract

The use of a committee of dialogue policies has been shown to be particularly beneficial for adaptation in multi-domain dialogue systems. In this model, each domain is represented by a committee member and the committee collaboratively decides which action the system should take. The action selection is optimised via a reinforcement learning algorithm where the reward is given to the committee member that refers to the current dialogue domain. While such a framework has improved the learning rates compared to single policy models, the main drawback is the need to know which domain the dialogue is in. This is a problem in real-world situations where the domain is a priori unknown, the user request might fall under several domains and the user may switch domains within the same dialogue. In order to deal with this, we augment the policy committee framework using ideas from multi-agent learning. Simulation results and a real user trial show that it is possible to optimise the committee decisions without prior knowledge of the domain.

1 Introduction

Statistical approaches to dialogue management have been shown to reduce design costs and provide superior performance to hand-crafted systems particularly in noisy environments [1]. Traditionally, spoken dialogue systems were built for limited domains described by an underlying ontology, which is essentially a structured representation of the database of entities that the dialogue system can talk about.

The semantic web is an effort to organise a large amount of information available on the Internet into a structure that can be more easily processed by a machine designed to perform reasoning about this data [2]. Knowledge graphs are one instance of such structures. They typically consist of a set of triples, where each triple represents two entities connected by a specified relationship. Current knowledge graphs have millions of entities and billions of relations and are constantly growing. There has been a significant amount of work in spoken language understanding focused on exploiting knowledge graphs in order to improve coverage [3][4]. More recently there have been some initial attempts to build statistical dialogue systems that operate on large knowledge graphs, but limited so far to the problem of belief tracking [5]. In this work, we address the problem of decision-making.

Moving from a limited domain dialogue system that operates on a relatively modest ontology to an open domain dialogue system that can converse about anything in a large knowledge graph is a non-trivial problem. An open domain dialogue system can be seen as a (large) set of limited domain dialogue systems. If each of them were trained separately then an operational system would require sufficient training data for every possible topic in the knowledge graph, which is simply not feasible. What is more likely is that instead we have limited and varied data drawn from different domains.
The architecture of a statistical dialogue system typically provides a single policy model that proposes actions throughout the dialogue [1]. This was initially also the case for multi-domain systems [6]. Multi-policy models have been proposed in the context of hierarchical modelling, where the decision-making process follows a hierarchy of policies, but at any given time only one policy makes a decision [7]. Concurrent policy models have been studied in [8] where several policies can propose an action at any given time and heuristics are used to decide which policy should be followed. Combining outputs of multiple policies was previously studied in the context of combining statistical and hand-crafted policies [9, 10]. Previous work on multi-domain dialogue systems has proposed a distributed architecture where a generic policy can be trained on data coming from different domains and later specialised to provide in-domain performance once sufficient data becomes available [11]. A policy committee model has been proposed [12] based on a Bayesian committee machine (BCM) [13]. It consists of a number of policies trained on different, potentially small, datasets. At any given time, when the system needs to make a decision, it consults each committee member and they each propose an action. A data-driven combination method is then used to reach the consensus.

While such a framework has improved the learning rates, the main drawback is that the dialogue manager needs to know which domain the dialogue is in. This is a problem in real world situations where the domain is a priori unknown, the user request might fall under several domains and the user may switch domains within the same dialogue. In order to deal with this, we augment the policy committee framework using ideas from multi-agent learning and show that it is possible to optimise the committee decisions without prior knowledge of the domain.

The rest of the paper is organised as follows. In Sections 2 and 3 we review Gaussian process reinforcement learning and the Bayesian committee machine respectively. Following that, in Section 4 we describe how multi-agent learning can be applied to the policy committee model. Section 5 presents the experimental set-up, followed by an evaluation of several committee combination models using a simulated user (Section 6) and real users (Section 7). We conclude the paper in Section 8 with a summary and future work directions.

2 Gaussian process 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 \( b \in B \), is estimated. The quality of a dialogue is defined by a reward function \( r(b, a) \) and the role of a dialogue policy \( \pi \) is to map the belief state \( b \) into a system action, an element of action space \( a \in A \), at each turn so as to maximise the expected cumulative reward.

The expected cumulative reward for a given belief state \( b \) and action \( a \) is defined by the \( Q \)-function:

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

(1)

where \( r_\tau \) is the immediate reward obtained at time \( \tau \), \( T \) is the dialogue length and \( \gamma \) is a discount factor, \( 0 < \gamma \leq 1 \). Optimising the \( Q \)-function is then equivalent to optimising the policy \( \pi \).

GP-Sarsa is an on-line reinforcement learning algorithm that models the \( Q \)-function as a Gaussian process [14]:

\[
Q(b, a) \sim GP(0, k((b, a), (b, a)))
\]  

(2)

where the kernel \( k(\cdot, \cdot) \) is factored into separate kernels over belief and action spaces \( k_B(b, b')k_A(a, a') \).

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

\[
Q(b, a) | r, B \sim N(Q(b, a), cov((b, a), (b, a)))
\]  

(3)
where the posterior mean and covariance take the form:

\[
\overline{Q}(b, a) = k(b, a)^T H^T (HH^T + \sigma^2 H^T)^{-1} r,
\]

\[
cov((b, a), (b, a)) = k((b, a), (b, a)) - k(b, a)^T H^T (HH^T + \sigma^2 H^T)^{-1} H k(b, a)
\]  \hspace{1cm} (4)

where \( k(b, a) = [k((b^0, a^0), (b, a)), \ldots, k((b^i, a^i), (b, a))]' \), \( H \) is the Gram matrix \([15]\), \( H \) is a band matrix with diagonal \([1, -\gamma]\) and \( \sigma^2 \) is an additive noise factor which controls how much variability in the \( Q \)-function estimate is expected during the learning process. Since the Gaussian process for the \( Q \)-function defines a Gaussian distribution for every belief state-action pair \([3]\), when a new belief point \( b \) is encountered, for each action \( a \in A \), there is a Gaussian distribution over \( Q \)-values. Sampling from these Gaussian distributions gives \( Q \)-values \( \tilde{Q}(b, a) \sim \mathcal{N}(\overline{Q}(b, a), \Sigma^Q(b, a)) \) where \( \Sigma^Q(b, a) = \text{cov}((b, a), (b, a)) \) from which the action with the highest sampled \( Q \)-value can be selected:

\[
\pi(b) = \arg \max_a \left\{ \tilde{Q}(b, a) : a \in A \right\}.
\]  \hspace{1cm} (5)

To use GPRL for dialogue, a kernel function must be defined on both the belief state space \( B \) and the action space \( A \). Here we use the Bayesian Update of Dialogue State (BUDS) dialogue model \([16]\). 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:

\[
k_A(a, a') = \delta_a(a')
\]  \hspace{1cm} (6)

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 user goal for each 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 \([17]\), which is a simple linear inner product. The kernel function for belief space is then the sum over all the individual hidden node kernels:

\[
k_B(b, b') = \sum_h \langle b_h, b'_h \rangle
\]  \hspace{1cm} (7)

where \( b_h \) is the probability distribution encoded in the \( h \)th hidden node.

### 3 Bayesian Committee Machine

The Bayesian committee machine is an approach to combining estimators that have been trained on different datasets and can be applied to Gaussian process regression \([13]\). Here we apply this method to combine the outputs of multiple estimates of \( Q \)-values \( Q_i \) with mean \( \overline{Q}_i \) and covariance \( \Sigma^Q_i \), given by Eq. (4) and trained on a set of rewards and belief-state action pairs \( r_i, B_i \) for \( i \in \{1, \ldots, M\} \), where \( M \) is the number of policies in the policy committee.

Following the description in \([13]\), the combined mean \( \overline{Q} \) and covariance \( \Sigma^Q \) are calculated as:

\[
\overline{Q}(b, a) = \Sigma^Q(b, a) \sum_{i=1}^M \Sigma^Q_i(b, a)^{-1} \overline{Q}_i(b, a),
\]

\[
\Sigma^Q(b, a)^{-1} = -(M-1) * k((b, a), (b, a))^{-1} + \sum_{i=1}^M \Sigma^Q_i(b, a)^{-1}.
\]  \hspace{1cm} (8)

### 4 Multi-agent learning in the policy committee framework

In the standard reinforcement learning framework there is a singe agent that is trying to solve one task in the given environment. However, for complex problems it has been shown \([18]\) that it is more
effective to decompose the problem into subproblems and have a system with multiple modules
where each module is trying to solve a subtask. Therefore, each module takes into account only a
part of the state space. This can significantly speed up the learning process. The learning in a multi-
agent system is typically performed in thee steps [18]. First, each agent proposes an action. Second,
a gating mechanism is deployed which selects the resulting system action. This mechanism can be
either handcrafted or optimised automatically. Finally, the reward that the system gets is distributed
among the agents and they each re-estimate their policy.

As can be seen, the multi-agent framework is closely related to the policy committee model. In fact,
the first two steps are exactly the same: each committee member estimates its own Q-function and
then Eq. 8 is used as the gating to automatically combine the output. In this work, we include the
third step, which is distributing the reward so that each committee member can learn from every
dialogue. So far, the total reward was given only to the committee member that is estimating the Q-
function for the current dialogue domain [12]. In practice it may be difficult to know which domain
the dialogue is currently in, the same user request may relate to different domains and the user can
switch domains within the same dialogue. Therefore we adopt three strategies for distributing the
reward:

 naïve approach: the total reward that the system obtains is directly fed back to each committee
member [13]

 winner-takes-all approach: the total reward that the system obtains is fed back to the committee
member that proposed the highest Q-value for the action that was finally chosen by the
gating mechanism [19]

 reward scaling approach: the total reward is redistributed to each committee member in such way
to reflect its contribution to the final action chosen by the gating mechanism [18]

5 Experimental set-up

In order to examine the ability of the proposed method to operate on multiple domains we examine
four domains: SFR consisting of restaurants in San Francisco, SFH consisting of hotels in San
Francisco, L11 consisting of laptops with 11 properties that the user can specify. In that way not
only the slots are different, but one of domains, L11, has many more slots than others. A description
of each domain with slots is given in [12].

6 Simulation results

In order to investigate the effectiveness of multi-agent learning within the policy committee model,
a variety of contrasts were examined using an agenda-based simulated user operating at the dialogue
act level [20]. 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.

The contrasts studied were as follows:

NAÏV Naïve approach – The total reward was given to each committee member.

WINN Winner-takes-all approach – The total reward was given to the policy member which on
average was giving the highest Q-value Q-variance ratio, \( \sum_i (b, a)^{-1} Q_i(b, a) \) from Eq. 8
for the action that was taken by the system.

SELFSCALE Scale received reward according to self Q-value estimate – Each policy com-
mittee member scales the total reward according to the average portion of its associated
Q-value for the action that the system took as opposed to other actions it proposed.

COMMSCALE Scale received reward according to all committee members’ Q-value estimate
– Each policy committee member scales the total reward according to its average portion
of the Q-value Q-variance ratio, \( \sum_i (b, a)^{-1} Q_i(b, a) \) from Eq. 8 for the action that the
system took as opposed to the Q-value Q-variance ratio that other committee members
proposed for the taken action.
For each method described above, 10 policies were trained on the simulated user using different considered multi-domain system for SFR, SFH and L11.

When there is more training data available, the winner-takes-all approach has better or equal performance to the approaches which distribute the reward. This is in line with the intuition that with abundance of data, the accuracy of the given reward is more important than the size of training data. If we average results across the domains and the sizes of the training data, we can see that it is still more effective to use the approaches which distribute the reward.

Table 1: Comparison of strategies for multi-domain adaptation. In-domain performance is measured in terms of reward, success rate and the average number of turns per dialogue. Results are given with 95% confidence intervals.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Reward</th>
<th>Success</th>
<th>#Turns</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFR trained on 750 dialogues from SFR, SFH, L11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAIV</td>
<td>7.00 ± 0.20</td>
<td>73.66 ± 0.86</td>
<td>7.70 ± 0.08</td>
</tr>
<tr>
<td>WINN</td>
<td>6.84 ± 0.21</td>
<td>75.81 ± 0.84</td>
<td>8.29 ± 0.09</td>
</tr>
<tr>
<td>SELFScale</td>
<td>6.86 ± 0.20</td>
<td>72.90 ± 0.87</td>
<td>7.68 ± 0.08</td>
</tr>
<tr>
<td>COMMSCALE</td>
<td>7.06 ± 0.21</td>
<td>75.29 ± 0.85</td>
<td>7.98 ± 0.09</td>
</tr>
<tr>
<td>MBCM</td>
<td>7.37 ± 0.20</td>
<td>76.60 ± 0.83</td>
<td>7.92 ± 0.08</td>
</tr>
<tr>
<td>SFR trained on 7500 dialogues from SFR, SFH, L11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAIV</td>
<td>9.45 ± 0.22</td>
<td>87.98 ± 0.85</td>
<td>8.14 ± 0.11</td>
</tr>
<tr>
<td>WINN</td>
<td>9.67 ± 0.18</td>
<td>89.24 ± 0.68</td>
<td>8.15 ± 0.09</td>
</tr>
<tr>
<td>COMMSCALE</td>
<td>9.41 ± 0.17</td>
<td>88.08 ± 0.66</td>
<td>8.18 ± 0.09</td>
</tr>
<tr>
<td>MBCM</td>
<td>9.67 ± 0.17</td>
<td>88.28 ± 0.66</td>
<td>7.96 ± 0.08</td>
</tr>
<tr>
<td>L11 trained on 750 dialogues from SFR, SFH, L11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAIV</td>
<td>8.82 ± 0.20</td>
<td>77.40 ± 0.82</td>
<td>6.63 ± 0.07</td>
</tr>
<tr>
<td>WINN</td>
<td>7.23 ± 0.22</td>
<td>72.35 ± 0.88</td>
<td>7.20 ± 0.09</td>
</tr>
<tr>
<td>SELFScale</td>
<td>7.69 ± 0.21</td>
<td>72.17 ± 0.88</td>
<td>6.72 ± 0.07</td>
</tr>
<tr>
<td>COMMSCALE</td>
<td>8.11 ± 0.21</td>
<td>74.61 ± 0.85</td>
<td>6.78 ± 0.08</td>
</tr>
<tr>
<td>MBCM</td>
<td>8.52 ± 0.20</td>
<td>77.09 ± 0.82</td>
<td>6.88 ± 0.07</td>
</tr>
<tr>
<td>L11 trained on 7500 dialogues from SFR, SFH, L11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAIV</td>
<td>10.92 ± 0.16</td>
<td>86.80 ± 0.70</td>
<td>6.42 ± 0.07</td>
</tr>
<tr>
<td>WINN</td>
<td>11.25 ± 0.18</td>
<td>88.51 ± 0.76</td>
<td>6.43 ± 0.08</td>
</tr>
<tr>
<td>COMMSCALE</td>
<td>11.24 ± 0.17</td>
<td>88.55 ± 0.69</td>
<td>6.44 ± 0.07</td>
</tr>
<tr>
<td>MBCM</td>
<td>10.73 ± 0.16</td>
<td>87.23 ± 0.66</td>
<td>6.70 ± 0.07</td>
</tr>
<tr>
<td>Averaged across domains and size of training data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAIV</td>
<td>8.94 ± 0.10</td>
<td>80.49 ± 0.42</td>
<td>7.31 ± 0.04</td>
</tr>
<tr>
<td>WINN</td>
<td>8.46 ± 0.10</td>
<td>80.37 ± 0.42</td>
<td>7.58 ± 0.05</td>
</tr>
<tr>
<td>COMMSCALE</td>
<td>8.83 ± 0.10</td>
<td>81.17 ± 0.40</td>
<td>7.38 ± 0.04</td>
</tr>
<tr>
<td>MBCM</td>
<td>9.06 ± 0.09</td>
<td>82.17 ± 0.38</td>
<td>7.35 ± 0.04</td>
</tr>
</tbody>
</table>

MBCM Multi-policy Bayesian committee machine – Each committee member is trained only on in-domain data, so the reward is passed only to the committee member which is dedicated for that domain. This method was introduced in [12] and requires the knowledge of the domain.

We examine two cases: when the training data is limited, with only 250 dialogues available for each domain, and when there is more training data available, 2500 for each domain. Similar to [12], we consider a multi-domain system for SFR, SFH and L11.

For each method described above, 10 policies were trained on the simulated user using different random seeds. Each policy was then evaluated using 1000 dialogues on each domain. The overall average reward, success rate and number of turns are given in Table 1 together with 95% confidence intervals.

There are few important conclusions to be drawn from the results. First, on a smaller dataset the approach which chooses the winner committee member to pass the total reward to is less effective than the approaches which distribute the reward. This is expected, as in the latter case the policy learns from a larger poll of dialogues, which is particularly useful in early stages of the optimisation process. Out of all methods that distribute the reward, the method which uses self-scaling has the poorest performance, so its performance is not examined on a larger dataset. On larger datasets, the winner-takes-all approach has better or equal performance to the approaches which distribute the reward. This is in line with the intuition that with abundance of data, the accuracy of the given reward is more important than the size of training data. If we average results across the domains and the sizes of the training data, we can see that it is still more effective to use the approaches which distribute the reward.
7 Real user evaluation

To fully examine the effectiveness of the proposed reward distribution approach, two set-ups were also trained in direct interaction with human users. First, a multi-policy Bayesian committee machine (MBCM) is trained from scratch using data from the SFR, SFH and L6\textsuperscript{1} domains, as presented in [11]. This policy committee operate on all three domains but is dependent on the knowledge of the domain for policy update. We compare this to the committee reward scaling (COMMSCALE), presented in Section 6, which distributes the reward to each committee member for each dialogue. We deployed the system in a telephone-based set-up, with subjects recruited via Amazon MTurk, in the same set up as in [11]. A recurrent neural network model was used to estimate the reward [21].

Fig. 1 shows the moving average reward as a function of the number of training dialogues for the L6 domain comparing the MBCM and COMMSCALE committee approaches. The committees were also trained on SFR and SFH domains in parallel. The training data across the domains was equally distributed. 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. As can be seen, the results confirm that it is not necessary for the committee to be aware of the domain, on the contrary, distributing reward to each committee member according to their contribution can even produce better performance than only sending the reward signal to the committee member dedicated for the current domain.

8 Conclusions and Future work

We have extended the policy committee model using ideas from multi-agent learning to distribute the reward signal among the committee members, making this model particularly useful in a real-world scenario where the domain is a priori unknown. In simulations, the proposed approach achieves a performance close to that which relies on domain information to assign the reward to the appropriate committee member, while in a real human trial, it produced better performance.

In the work presented here, it is only the action-selection component does not use any domain information. Choosing which domain information to add to the chosen action, as well as the spoken language understanding, requires the knowledge of the domain. In future, we plan to apply this method in combination with a domain or a topic tracker operating over a large knowledge graph, with the aim of building open domain statistical dialogue systems.

Acknowledgments

The research leading to this work was funded by the EPSRC grant EP/M018946/1 Open Domain Statistical Spoken Dialogue Systems.

\textsuperscript{1}L6 is a laptop domain which contains 6 attributes that the user can specify.
References


