Reward Estimation for Dialogue Policy Optimisation

Pei-Hao Su, Milica Gašić and Steve Young

Department of Engineering, University of Cambridge, Cambridge, UK

Abstract

Viewing dialogue management as a reinforcement learning task enables a system to learn to act optimally by maximising a reward function. This reward function is designed to induce the system behaviour required for the target application and for goal-oriented applications, this usually means fulfilling the user's goal as efficiently as possible. However, in real-world spoken dialogue system applications, the reward is hard to measure because the user's goal is frequently known only to the user. Of course, the system can ask the user if the goal has been satisfied but this can be intrusive. Furthermore, in practice, the accuracy of the user's response has been found to be highly variable. This paper presents two approaches to tackling this problem. Firstly, a recurrent neural network is utilised as a task success predictor which is pre-trained from off-line data to estimate task success during subsequent on-line dialogue policy learning. Secondly, an on-line learning framework is described whereby a dialogue policy is jointly trained alongside a reward function modelled as a Gaussian process with active learning. This Gaussian process operates on a fixed dimension embedding which encodes each varying length dialogue. This dialogue embedding is generated in both a supervised and unsupervised fashion using different variants of a recurrent neural network. The experimental results demonstrate the effectiveness of both off-line and on-line methods. These methods enable practical on-line training of dialogue policies in real-world applications.

Keywords: Dialogue systems, Reinforcement learning, Deep learning, Reward Estimation, Gaussian process, Active learning

Email address: `phs26cam.ac.uk, mg436@cam.ac.uk, sjy11@cam.ac.uk` (Pei-Hao Su, Milica Gašić and Steve Young)

1. Introduction

Spoken Dialogue Systems (SDS) allow human-computer interaction using natural speech. They can be broadly divided into two categories: chat-based systems which converse with users with the aim of providing contextually relevant responses in broad domains [1, 2], and task-oriented systems designed to assist users to achieve specific goals (e.g. find hotels, movies or bus schedules) [3]. The latter are typically designed on top of a structured *ontology* (or a database *schema*), which defines the domain that the system can talk about. The development of such systems traditionally requires a substantial amount of hand-crafted rules combined with various statistical components. These include a spoken language understanding module [4, 5], a dialogue belief state tracker [6] to predict user intents and track the dialogue history, a dialogue policy [7] to determine the dialogue flow, and a natural language generator [8] to convert abstract system responses into natural language. Teaching such a system how to respond appropriately in all situations is non-trivial. Traditionally, the *dialogue management* component has been designed manually using flow charts to directly specify system behaviour. More recently, it has been formulated as a planning problem and solved using reinforcement learning (RL) to enable automatic optimisation during interaction with users [7, 9, 10, 11, 12, 13, 14, 15, 16]. In this framework, the system learns by a *trial and error* process governed by a potentially delayed learning objective, a *reward function*.

A typical approach to defining the reward function in a task-oriented dialogue system is to apply a small per-turn penalty to encourage short dialogues and to give a large positive reward at the end of each successful interaction. Figure 1 is an example of a dialogue task which is typically set when bootstrapping or evaluating a system using paid subjects. When paid subjects are primed with a specific task to complete, dialogue success can be determined from both subjective user ratings (*Subj*), and from an objective measure (*Obj*) based on whether or not the pre-specified task was completed [17, 18]. However, when operating with real users, prior knowledge of the user's goal is not normally available and hence it is not possible to compute an objective reward.

Furthermore, objective ratings must necessarily be quite strict to ensure that paid subjects properly exercise the system. As a consequence, tasks often fail because the paid subject forgot to ask for a required piece of information. For example, in Figure 1, the paid subject forgot to ask for the

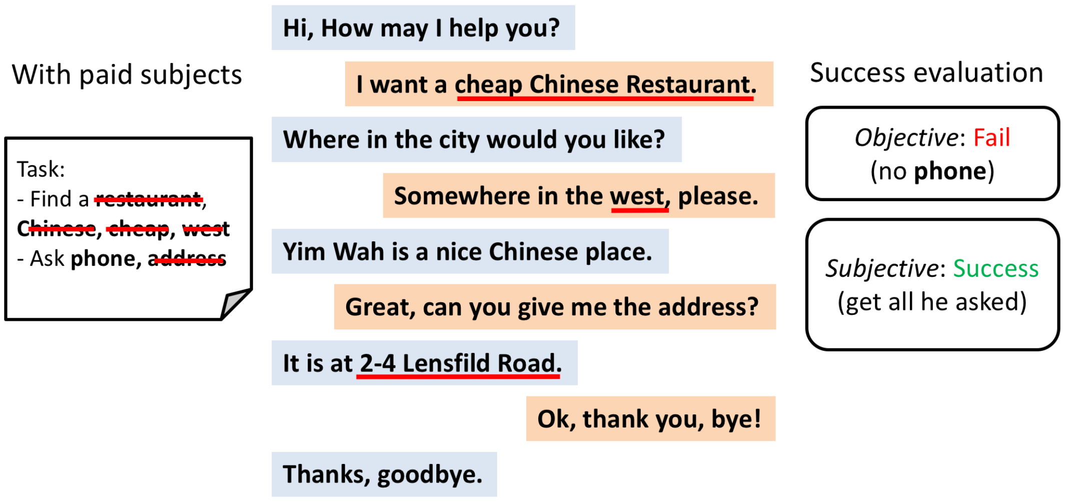

With paid subjects
Task:
- Find a ~~restaurant~~,
~~Chinese~~, ~~cheap~~, ~~west~~
- Ask **phone**, ~~address~~
Hi, How may I help you?
I want a cheap Chinese Restaurant.
Where in the city would you like?
Somewhere in the west, please.
Yim Wah is a nice Chinese place.
Great, can you give me the address?
It is at 2-4 Lensfild Road.
Ok, thank you, bye!
Thanks, goodbye.
Success evaluation
Objective: Fail
(no **phone**)
Subjective: Success
(get all he asked)

Figure 1: An example of a task-oriented dialogue with a pre-defined task and the evaluation results.

phone number. This results in a mismatch between the *Obj* and *Subj* ratings.

Unfortunately, relying on subjective ratings alone is also problematic since paid subjects frequently give inaccurate responses partly because they forget the complete task as in the example above, and also because they are concerned that their answers will affect their payment. Real users are often unwilling to extend the interaction in order to give feedback, and even when they do, their feedback can also be unreliable [18], for example due to sociological effects such as not wishing to be impolite.

All of the above can result in unstable learning [19, 20]. When bootstrapping a system using paid users, an effective albeit inefficient solution is to ignore all dialogues for which the objective success assessment differs from the subjective success assessment (referred to as the $Obj = Subj$ check below) [18]. However, as well as being inefficient, this solution does not help address the real problem which is how to train systems on-line with real users when the user's goal is generally unknown and difficult to infer.

To deal with the above issues, this paper investigates the use of an independent reward estimator which can be trained to monitor a dialogue and accurately estimate task success independently of the specific goal set, or of whatever goal is in the user's mind. The investigation is in two parts.

We first describe a recurrent neural network (RNN) designed to estimate objective success *Obj*, which is trained from off-line simulated dialogue data [21, 22]. The resulting policy is found to be as effective as the one trained with dialogues using the $Obj = Subj$ check. However, a user simulator will only provide a rough approximation of real user statistics and developing a user simulator is a costly process [23]. In addition, the interactions encountered during subsequent deployment might be distributed very differently to those generated by the simulator.

To counter this, the second part of the paper describes an on-line active learning method in which users are asked to provide feedback on whether the dialogue was successful or not [24]. In this set-up, active learning is used to limit requests for user feedback to only those cases where the feedback would be useful. The estimator used is a Gaussian process classification (GPC) model. This has the advantage that as well as being highly data efficient and providing a measure of uncertainty in its estimates, it also incorporates a noise model which can be used to compensate for cases where the user feedback is inaccurate. Since GPC operates on a fixed-length observation space and dialogues are of variable-length, an RNN-based embedding function is used to provide fixed-length dialogue representations.

When the above components are integrated with a GP-based dialogue management policy, we demonstrate that both the dialogue policy and the reward estimator can simultaneously learn on-line from scratch, making this framework directly applicable to real-world applications in which systems learn directly in interaction with real users performing real-world tasks.

The rest of the paper is organised as follows. The following section gives an overview of related work and then Section 3 introduces the components of our spoken dialogue system, the operating domain, and the learning algorithm used for dialogue policy training. The proposed off-line reward estimator is presented in Section 4, where the turn-level dialogue features and the off-line reward model are described. Following this, the framework of on-line active reward estimation is introduced in Section 5. This includes the design of the dialogue embedding function and the active reward model trained from real user feedback. In Section 6, the proposed approaches are evaluated in the context of an application providing restaurant information in Cambridge, UK. The accuracy of the off-line reward model on a reward prediction task is given and then the results of dialogue policy learning are presented. To provide some insight into the effectiveness of the dialogue embedding, an analysis of the dialogue embedding space is also presented. Finally, the performance of the active reward model when simultaneously trained with the dialogue policy on-line with real users is presented. Concluding remarks are made in Section 7.

2. Related Work

Dialogue evaluation has been an active research area since late 90s. In the PARADISE framework, a linear function of task completion and various other dialogue features such as dialogue duration were used to infer user satisfaction[17]. This measure was later used as a reward function for learning a dialogue policy [25]. However, as noted above, obtaining an accurate estimate of task completion is difficult when the system is interacting with real users. Also, concerns have been raised regarding the theoretical validity of the PARADISE model [26].

Several approaches have been adopted for learning a dialogue reward model given a corpus of annotated dialogues. For example, Yang et al used collaborative filtering to infer user preferences[27]. The use of reward shaping has also been investigated to enrich the reward function in order to speed up dialogue policy learning[28, 29]. It has also been demonstrated that there is a strong correlation between an expert's user satisfaction rating and dialogue success [30]. However, all these methods assume the availability of accurate dialogue annotations such as expert ratings, which in practice are hard to obtain.

One effective way to mitigate the effects of annotator error in lower quality annotations is to obtain multiple ratings for the same data and several methods have been developed to guide the annotation process using uncertainty models [31, 32]. Active learning is particularly useful for determining when an annotation is needed [33, 34] and it is often implemented using Bayesian optimisation [35]. For example, Daniel et al exploited a pool-based active learning method for a robotics application[36]. They queried the user for feedback on the most informative sample collected so far and showed the effectiveness of this method. Active learning coupled with Gaussian process regression has been previously used for dialogue policy optimisation and shown to be more sample efficient than alternative methods [7].

Rather than explicitly defining a reward function, inverse RL (IRL) aims to recover the underlying reward from demonstrations of good behaviour and then learn a policy which maximises the recovered reward [37]. IRL was first introduced into SDS design by Paek et al, where the reward was inferred from human-human dialogues to mimic the behaviour observed in a corpus [38]. IRL has also been studied in a Wizard-of-Oz (WoZ) setting [39, 40], where typically a human expert served as the dialogue manager to select each system reply based on the speech understanding output at different noise levels. However, this approach is costly and there is no reason to suppose that a human wizard is acting optimally, especially at high noise levels.

Since humans are better at giving relative judgements than absolute scores, another related line of research has focused on preference-based approaches to RL [41]. For example, in one study, users were asked to provide rankings between pairs of dialogues [42]. However, this is also costly and does not scale well in real applications.

3. The Core Spoken Dialogue System

The Cambridge restaurant domain is used for all of the experimental work described in this paper. Users converse with the system to find restaurants in Cambridge that match their required constraints such as food type, area, etc. The domain consists of approximately 150 venues and each venue has six attributes (slots) of which three (area, price-range and food-type) can be used by the system to constrain the search and the remaining three (phone number, address and postcode) are informable properties that can be queried by the user once a database entity has been found.

Two operating modes are used in the experiments: **live user trial mode** and **user simulation mode**. In live user trial mode, the core components of the SDS comprise a domain independent HMM-based speech recogniser, a confusion network (CNet) semantic decoder [4], the BUDS belief state tracker that factorises the dialogue state using a dynamic Bayesian network[43], a Gaussian process-based dialogue manager [7], and a template-based natural language generator (NLG) to map system actions into natural language responses back to the user.

In user simulation mode, an agenda-based simulated user is used to interact with the system at the abstract dialogue act level [44]. In this mode, the system consists of only the BUDS belief state tracker and the dialogue manager. The user simulator includes an error generator, which can be set to generate user inputs with different semantic error rates (SER).

The core of the dialogue manager is a dialogue policy π which maps the *belief state* $\mathbf{b} \in \mathcal{B}$, a distribution over all possible dialogue states at each turn, to a system action $a \in \mathcal{A}$, which in live mode is converted into natural language using the NLG component.

As explained in the introduction, the behaviour of a statistical dialogue system is conditioned by a reward function $r(\mathbf{b}, a)$ which for any given belief state $\mathbf{b}$ and action a defines the *immediate reward* for that turn. Dialogue policy optimisation seeks to use reinforcement learning to maximise the total reward expected over each complete dialogue.

An estimator for the expected cumulative reward (also called the *return*) from any given belief state $\mathbf{b}$ and action a to the end of the dialogue is often defined by a Q-function:

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

where r_τ is the immediate reward obtained at time τ and T is the dialogue length[1]. Optimising the Q-function is equivalent to optimising the policy π.

For all of the experiments described in this paper, the Q-function is modelled by a Gaussian process (GP):

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

where $m(\cdot, \cdot)$ is the prior mean function and the kernel $k(\cdot, \cdot)$ is factored into separate kernels over belief and action spaces $k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}')k_{\mathcal{A}}(a, a')$. The policy is optimised using an algorithm called GP-SARSA [7, 45] in which the Q-function is updated by calculating the posterior given the collected belief-action pairs $(\mathbf{b}, a)$ (dictionary points) and their corresponding rewards.

GP-based reinforcement learning (GPRL) is particularly appealing since it can learn from a small number of observations by exploiting the correlations defined by the kernel function and, as a bonus, it also provides a measure of the uncertainty in its estimates. This knowledge of the distance between data points in the observation space greatly speeds up policy learning since the Q-values of the unexplored space can be estimated from the Q-values of nearby points. However, computation and model complexity become intractable if every data point had to be memorised. So instead, sparse approximation

[1]The total reward in RL often includes a discount factor to discount the value of future rewards. Dialogues in a task oriented SDS typically require only 6 or 7 turns, and they rarely extend beyond 20 turns. Hence the discount factor is normally set to 0.99 or 1.0.

methods such as *kernel span* [45] are used to bound the size of stored training points.

Further reductions in data space requirements can be achieved by condensing the action set into a fixed set of *summary* actions, which are then heuristically mapped back into the *master* system action space using a set of hand-crafted rules. The summary action space consists of a set of slot-dependent and slot-independent summary actions. Slot-dependent summary actions include requesting the slot value, confirming the most likely slot value and selecting between the top two slot values.

The summary action kernel is defined as:

$$k_{\mathcal{A}}(a, a') = \delta_a(a') \tag{3}$$

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 [46], which is a simple linear inner product. The kernel function for belief space is then the sum over all the individual hidden node kernels:

$$k_{\mathcal{B}}(\mathbf{b}, \mathbf{b}') = \sum_{\mathbf{h}} \langle \mathbf{b}_{\mathbf{h}}, \mathbf{b}'_{\mathbf{h}} \rangle \tag{4}$$

where $\mathbf{b}_{\mathbf{h}}$ is the probability distribution encoded in the h^{th} hidden node.

4. Off-line Reward Estimation

In this section, a recurrent network-based reward estimator is described which can be trained to estimate task success based on turn-level features extracted directly from each dialogue. Once trained, the reward estimator enables on-line policy optimisation with real users without access to either *Obj* or *Subj* task success ratings. It does however require training data annotated for task success which in the work described here is generated off-line by a user simulator.

Recurrent neural networks (RNNs) are a subclass of neural network models with recurrent connections from one time-step to the next. Their ability to succinctly capture and retain history information makes them suitable for modelling sequential data with temporal dependencies. They have been previously used with success in a variety of natural language processing (NLP) tasks such as language modelling [47, 48, 49] and spoken language understanding (SLU) [50].

4.1. Training data and dialogue features

The data used to train all of the reward estimation models was collected by training several Gaussian process policies [7] at various error rates (see Section 6.2 below) from scratch using an agenda-based simulated user [44]. Each dialogue was labelled as a success or failure using the objective criteria (*Obj*) described in the introduction i.e. a dialogue was labelled as successful if all of the users' goals randomly generated at the start of the dialogue were completed. The reward function used during policy training gave -1 at each turn to encourage brevity, and +20 at completion if the dialogue was successful, otherwise 0. The return (cumulative reward) $\Re$ was, therefore, calculated as:

$$\Re = 20 \times \mathbb{1}_{success} - N \tag{5}$$

where N is the number of dialogue turns and $\mathbb{1}_{success}$ is an indicator function for success.

For all models, a feature vector $\mathbf{f}_t$ was extracted at each turn t consisting of the following concatenated sections: a one-hot encoding of the user's top-ranked dialogue act, the real-valued belief state vector formed by concatenating the distributions over all goal (area, price range, food type), method and history variables [43], a one-hot encoding of the summary system action, and the current turn number (see Figure 2).

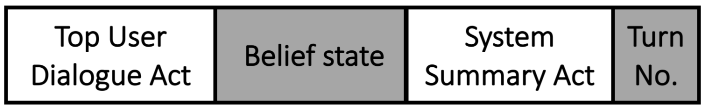


Figure 2: Feature vector $\mathbf{f}_t$ extracted at each turn t.

This form of feature vector was motivated by considering the primary information a human would require to read a transcription and rate the

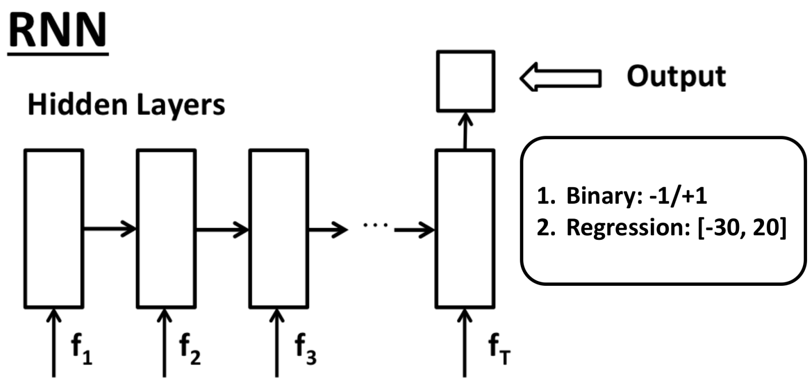


Figure 3: An unrolled view of the RNN model. The feature vectors extracted at turns $t = 1, \ldots, T$ are labelled $\mathbf{f}_t$. Two types of output are considered: binary classification and regression.

success of the dialogue. The inclusion of the belief vector, the user dialogue act and the system's actions makes this feature vector domain and system dependent.

The aim of the RNN reward predictor is to enable policy learning with real users without prior knowledge of the users' goal. To do this, the model should consider the information available at every turn of the dialogue and then evaluate whether or not the policy provided everything that was asked for. The hope is that by training the RNN model on data from simulated users, it will generalise and provide accurate assessments of dialogues with real users whose goals are not known *a priori*.

4.2. Recurrent Neural Network Structure

The RNN reward predictor takes as input the feature vectors $\mathbf{f}_t$ described above and updates its hidden layer $\mathbf{h}_t$ at each turn t.[2] Once the dialogue ends, the hidden layer is then connected to an output layer to make a single reward prediction for the whole dialogue as depicted in Figure 3.

The network structure in the final layer is determined by the choice of supervised training targets, of which two types were considered:

[2]Convolutional neural networks were also investigated on the same task as in [21] However, only results using an RNN are shown here since it was found to be generally more robust.

1) Classification model: in this case, the RNN is formulated as a binary classifier which is trained to predict success or failure for each dialogue. The targets are $\{0, 1\}$ and the final layer of the RNN outputs a scalar through a sigmoid activation function and is trained with a cross-entropy loss. The output from this network is a probability p that the dialogue is a success, and the hard class label predicted by the model is taken as 1 if $p > 0.5$, else 0. This hard label is used to determine whether to give a final reward of $+20$ during policy learning, as per Equation 5.

2) Regression model: in this case, the RNN is formulated as a regressor with the actual return value used as the training target. The model's final layer has no non-linearity (activation) and the whole model is trained with a mean-square-error (MSE) loss function. During subsequent policy learning using this predictor, the per-turn penalty is suppressed since it is already taken account of by the RNN reward predictor.

5. On-line Active Reward Learning

The off-line reward predictor described above suffers from the obvious problem that it requires either a user simulator to be built or a substantial amount of training data to be collected and annotated. Furthermore, whichever approach is taken, the training data may be a poor match to the user population used for on-line policy optimisation.

An alternative is to learn a predictor directly on the real user population in parallel with policy optimisation, with the label annotation provided by subjective user feedback. However as noted above, subjective user ratings, especially ones collected via crowd-sourcing, tend to be inaccurate [20], potentially intrusive and hard to collect since users will often just "hang-up" as soon as they have the information they need. In this section, a Gaussian process-based reward estimator is described which uses active learning to limit intrusive requests for feedback and a noise model to mitigate the effects of inaccurate feedback [24].

The proposed system framework is depicted in Figure 4. It is divided into two main parts: a dialogue embedding function, and an active reward model to obtain user feedback and predict dialogue success. When each dialogue ends, a sequence of turn-level features $\mathbf{f_t}$ as illustrated in Section 4.1 is extracted and fed into an embedding function ς to obtain a fixed-dimension dialogue representation $\mathbf{d}$ that serves as the input to the reward model. This reward is modelled as a Gaussian process which for every input $\mathbf{d}$ provides

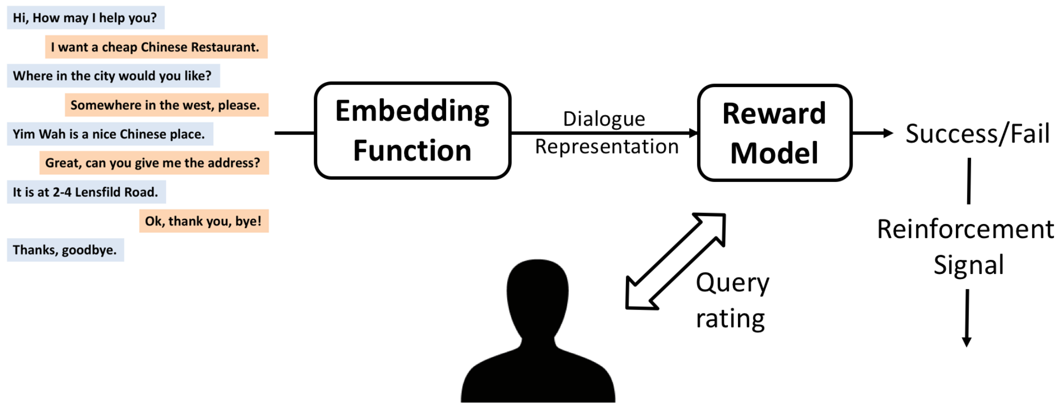

Hi, How may I help you?
I want a cheap Chinese Restaurant.
Where in the city would you like?
Somewhere in the west, please.
Yim Wah is a nice Chinese place.
Great, can you give me the address?
It is at 2-4 Lensfild Road.
Ok, thank you, bye!
Thanks, goodbye.
Embedding Function
Dialogue Representation
Reward Model
Success/Fail
Reinforcement Signal
Query rating

Figure 4: Schematic depiction of the on-line reward estimator. The two main system components: dialogue embedding creation, and reward modelling based on user feedback, are described in Section 5.

an estimate of task success along with an estimate of the uncertainty. Based on this uncertainty, the reward model decides whether to query the user for feedback or not. It then returns a reinforcement signal to update the dialogue policy π, which is trained using the GP-SARSA algorithm as described in Section 3.

Note that the reward model and the dialogue policy are being jointly optimised during the sequence of dialogues. Initially, the joint learning process is being supervised by the user feedback. Once learning is underway, the user supervision becomes lighter and ultimately the learning becomes essentially unsupervised.

5.1. Dialogue Embeddings

The use of embedding functions for sequence modelling has recently gained attention especially for word representations, and it has boosted performance on several natural language processing tasks [51, 52, 53]. Embedding has also been successfully applied to machine translation (MT) where it enables varying-length phrases to be mapped to fixed-length vectors using an RNN Encoder-Decoder [54]. Here an embedded representation of the sequence of dialogue turn level features is computed in order to drive the reward model.

Two methods of generating dialogue embeddings have been explored: a *supervised* and an *unsupervised* approach. Once computed, these embeddings are used as the observations for the reward model described in the section Section 5.2.

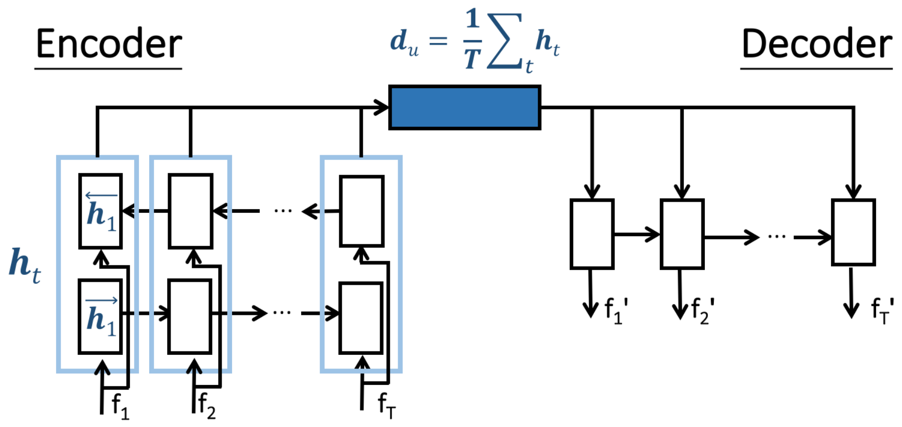


Figure 5: Unsupervised dialogue embedding using the LSTM Encoder-Decoder structure.

5.1.1. Supervised Dialogue Embedding

In this case, the RNN described in Section 4.2 is used whereby the hidden layer h_T at the final turn T serves as the dialogue representation $\mathbf{d_s}$ to summarise the entire dialogue in a single fixed-length vector. Note, however, that as with the off-line reward estimator described in 4.2, this approach requires training data which is expensive to collect and may be biased with respect to the target population.

5.1.2. Unsupervised Dialogue Embedding

To avoid the need for labelled training data, an encoder-decoder structure can be used to generate dialogue representations. The basic set-up is illustrated in Figure 5. Turn level feature sequences $\mathbf{f_t}$ as described in Section 4.1 are input to the model, encoded and then decoded to reconstruct the input. The encoder is a Bi-directional Long Short-Term Memory network (BLSTM) [55, 56], which has been shown to outperform uni-directional LSTM in various tasks [57, 58]. It takes into account the sequential information from both directions of the input data, computing the *forward* hidden sequences $\overrightarrow{\mathbf{h}}_{1:T}$ and the *backward* hidden sequences $\overleftarrow{\mathbf{h}}_{T:1}$ while iterating over all input features $\mathbf{f_t}$, $t = 1, ..., T$:

$$\overrightarrow{\mathbf{h}_\mathbf{t}} = LSTM(\mathbf{f_t}, \overrightarrow{\mathbf{h}}_{t-1})$$
$$\overleftarrow{\mathbf{h}_\mathbf{t}} = LSTM(\mathbf{f_t}, \overleftarrow{\mathbf{h}}_{t+1})$$

where $LSTM$ denotes the activation function. The dialogue representation $\mathbf{d_u}$ is then calculated as the average over all hidden sequences:

$$\mathbf{d_u} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{h_t} \tag{6}$$

where $\mathbf{h_t} = [\overrightarrow{\mathbf{h_t}}; \overleftarrow{\mathbf{h_t}}]$ is the concatenation of the two directional hidden sequences.[3] The decoder is a forward LSTM that takes $\mathbf{d_u}$ as its input for each turn t to produce the sequence of features $\mathbf{f'}_{1:T}$. The encoder-decoder is trained by minimising the mean-square-error (MSE) using stochastic gradient decent (SGD):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} ||\mathbf{f}_t - \mathbf{f}'_t||^2 \tag{7}$$

where N is the number of training dialogues and $|| \cdot ||^2$ denotes the l^2-norm.

5.2. Active Reward Learning

A Gaussian process is a Bayesian non-parametric model that can be used for regression or classification [59]. It is particularly appealing since it can learn from a small number of observations by exploiting the correlations defined by a *kernel function* and it provides a measure of uncertainty of its estimates. In the context of spoken dialogue systems it has been successfully used for RL policy optimisation [7, 60] and IRL reward function regression [61].

Here we propose modelling dialogue success as a Gaussian process (GP). This involves estimating the probability $p(y|\mathbf{d}, \mathcal{D})$ that the task was successful given the current dialogue representation $\mathbf{d}$ and the pool $\mathcal{D}$ containing previously classified dialogues. We pose this as a classification problem where the rating is a binary observation $y \in \{-1, 1\}$ that defines failure or success. The observations y are considered to be drawn from a Bernoulli distribution with a success probability $p(y = 1|\mathbf{d}, \mathcal{D})$. The probability is related to a latent function $f(\mathbf{d}|\mathcal{D}) : \mathcal{R}^{dim(\mathbf{d})} \to \mathcal{R}$ that is mapped to a unit interval by a *probit* function $p(y = 1|\mathbf{d}, \mathcal{D}) = \phi(f(\mathbf{d}|\mathcal{D}))$, where ϕ denotes the cumulative density function of the standard Gaussian distribution.

[3]Note use of a bi-directional LSTM does not cause any difficulty at run time because we are only interested at predicting success at the end of the dialogue.

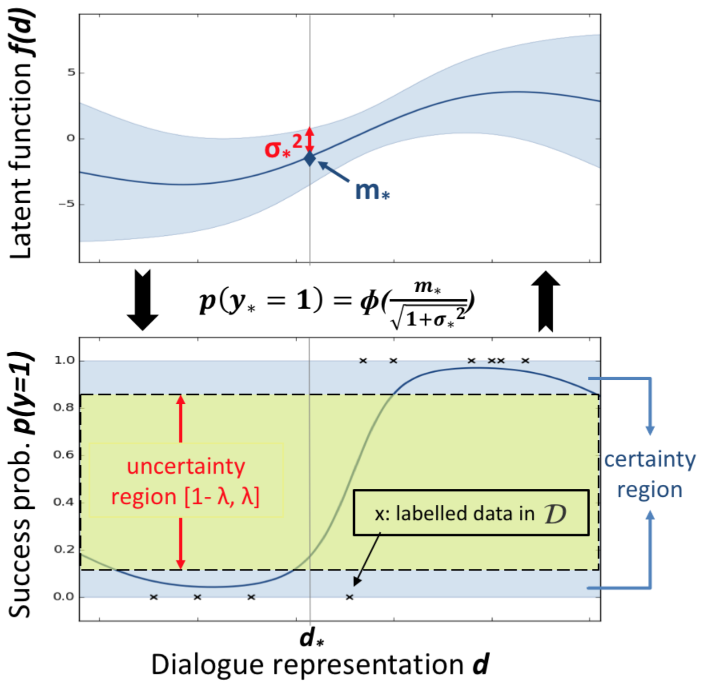


Figure 6: 1-dimensional example of the proposed GP active reward learning model.

The latent function is given a GP prior: $f(\mathbf{d}) \sim \mathcal{GP}(m(\mathbf{d}), k(\mathbf{d}, \mathbf{d}'))$, where $m(\cdot)$ is the mean function and $k(\cdot, \cdot)$ the covariance function (kernel). The stationary squared exponential kernel k_{SE} is used. It is also combined with a white noise kernel k_{WN} to account for the "noise" in users' ratings:

$$k(\mathbf{d}, \mathbf{d}') = p^2 \exp(-\frac{||\mathbf{d} - \mathbf{d}'||^2}{2l^2}) + \sigma_n^2 \tag{8}$$

where the first term denotes k_{SE} and the second term k_{WN}.

The *hyper-parameters* p, l, and σ_n can be optimised by maximising the marginal likelihood using a gradient-based method [62]. Since $\phi(\cdot)$ is not Gaussian, the resulting posterior probability $p(y = 1|\mathbf{d}, \mathcal{D})$ is analytically intractable. So instead approximate optimisation was performed using expectation propagation (EP) [63].

Querying the user for feedback is costly and may have a negative impact on the user experience. This can be reduced by using active learning based on uncertainty estimates of the GP model [64]. This ensures that user feedback is only sought when the model is uncertain about its current prediction. For the case here, an on-line (stream-based) version of active learning is required.

A 1-dimensional example is shown in Figure 6. Given the labelled data $\mathcal{D}$, the predictive posterior mean μ_* and posterior variance σ_*^2 of the latent value $f(\mathbf{d}_*)$ for the current dialogue representation $\mathbf{d}_*$ can be calculated. Then a threshold interval $[1-\lambda, \lambda]$ is set on the predictive success probability $p(y_* = 1|\mathbf{d}_*, \mathcal{D}) = \phi(\mu_*/\sqrt{1+\sigma_*^2})$ to decide whether this dialogue should be labelled or not. The decision boundary implicitly considers both the posterior mean as well as the variance.

When deploying this reward model in the proposed framework, a GP with a zero-mean prior for f is initialised and $\mathcal{D} = \{\}$. After the dialogue policy π completes each episode with the user, the generated dialogue turns are transformed into the dialogue representation $\mathbf{d} = \sigma(\mathbf{f}_{1:T})$ using the dialogue embedding function ς. Given $\mathbf{d}$, the predictive mean and variance of $f(\mathbf{d}|\mathcal{D})$ are determined, and the reward model decides whether or not it should seek user feedback based on the threshold λ on $\phi(f(\mathbf{d}|\mathcal{D}))$. If the model is uncertain, the user's feedback on the current episode $\mathbf{d}$ is used to update the GP model and to generate the reinforcement signal for training the policy π; otherwise the predictive success rating from the reward model is used directly to update the policy. This process takes place after each dialogue.

6. Experiments

6.1. Experimental Settings

All of the experiments used the target domain and spoken dialogue system structure described in Section 3. Policy training used the reward given by Equation 4.1 and the maximum dialogue length was set to 30. The on-line system used paid subjects recruited via Amazon Mechanical Turk (AMT). Each user was assigned specific tasks in a given sub-domain and then asked to call the system in a similar set-up to that described in [65, 66]. After each dialogue, the users were asked whether they judged the dialogue to be successful or not. Based on that binary rating, the subjective success was calculated as well as the average reward. An objective rating was also computed by comparing the system outputs with the assigned task specification.

6.2. Off-line Reward Estimator

6.2.1. RNN-based Dialogue Success Prediction

The off-line reward estimator described in Section 4 was implemented using the Theano library [67, 68], with a hidden layer of 300 units and sigmoid activation units. All models were trained using stochastic gradient descent (per dialogue) on data generated by three independently trained GP-based policies interacting with a simulated user at a 15% semantic error rate (SER) [69]. One model was fully trained on 18K dialogues (`train-18K`) and second model was trained on only 1K dialogues (`train-1K`). The contrast between the larger `train-18K` set and the smaller `train-1K` set is provided to give some insight into the impact of reduced availability of training data. A separate validation set of 1K dialogues was also generated to avoid overfitting.

There were two test sets: a matched set of 1K dialogues generated at SER 15% (`testA`) and a larger set containing 3K dialogues at each of four SERs: 0%, 15%, 30%, and 45% (`testB`) where the latter provides an indication of how models perform when there is a mis-match between train and test sets.

For each dialogue, the input features were represented as the concatenation of four segments as shown in Figure 2, which were realised in two configurations: a domain-dependent feature vector $\mathcal{F}_{617}$ and a domain-independent feature vector $\mathcal{F}_{74}$ [22] where the subscripts denote the size of the vectors.

For $\mathcal{F}_{617}$, the number of elements were 21, 575, 20, 1 respectively for the user act, full belief state, system action and rescaled turn number. The full belief state was the concatenation of the distribution over each domain-dependent slot such as price-range and area and the distribution over domain-independent features relating to the user's discourse. For the system act, a fixed set of generic system actions (e.g. goodbye) and N_a slot-dependent system actions was available for each slot $\mathcal{S}$ (e.g. '*request-$\mathcal{S}$*', '*confirm-$\mathcal{S}$*', etc.). Given N_s slots, this will result in $N_s \times N_a$ slot-dependent system actions.

For $\mathcal{F}_{74}$, the number of features were 21, 38, 14, 1 respectively for the user act, summarised belief state, domain-independent system action and rescaled turn number. Note that this was designed to be a light-weight feature vector applicable across multiple domains. The difference between the summarised and full belief state is that each set of domain-dependent slot distributions is represented by a normalised entropy value. Thus the entire summarised belief state is domain-independent. Similarly, a domain-independent system action was created by mapping slot-dependent system actions to a single slot-independent action (e.g. '*request-$\mathcal{S}_1$*', ..., '*request-$\mathcal{S}_{N_s}$*' $\mapsto$ '*request*'). The rescaled turn number was expressed as a percentage of the maximum number of allowed turns, here 30. The one-hot user dialogue action encoding was formed by taking only the most likely user action estimated by the CNet semantic decoder.

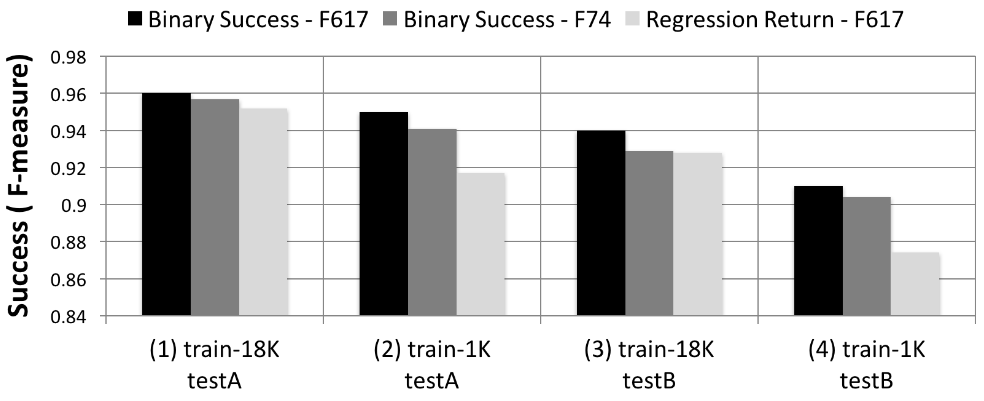


Figure 7: Prediction of the RNN model trained on 18K and 1K dialogues and tested on sets *testA* and *testB* (see text). Results of success/failure label F-measure (left axis) are represented as bars.

As described in Section 4.2, both binary success classification models and reward regression models were trained. For the case of binary success classification, both input feature sets $\mathcal{F}_{617}$ and $\mathcal{F}_{74}$ were tested. For the regression case only $\mathcal{F}_{617}$ was tested. The results are shown in Figure 7, where the y-axis is the F-measure of the success classification (bar plot). When using the large training set (18K, sub-figures 7(1) & 7(3)), all models obtained an F-score above 0.93 on success label prediction and were within $\sim \pm 0.03$ of the objective return targets on *testA* and within $\sim \pm 0.05$ on *testB*.

Without the benefit of a simulated user, it may not be possible to obtain sufficient labelled training dialogues to fully train the RNN model. However, the results shown in sub-figures 7(2) & 7(4) suggest that the model is reasonably robust even with as little as 1000 training dialogues. In this case, the binary classification model is the most accurate.

Note that the domain specific feature set $\mathcal{F}_{617}$ only slightly outperformed the domain independent $\mathcal{F}_{74}$ set. Therefore given the desirability of domain independence, the remaining sections focus on the use of the $\mathcal{F}_{74}$ feature set.

Overall, these results suggest that RNNs, sequentially processing turn level features, are able to provide useful estimates of success and reward. The results on set *testB* also show that the models can perform well under varying error rates as would be encountered in real operating environments.

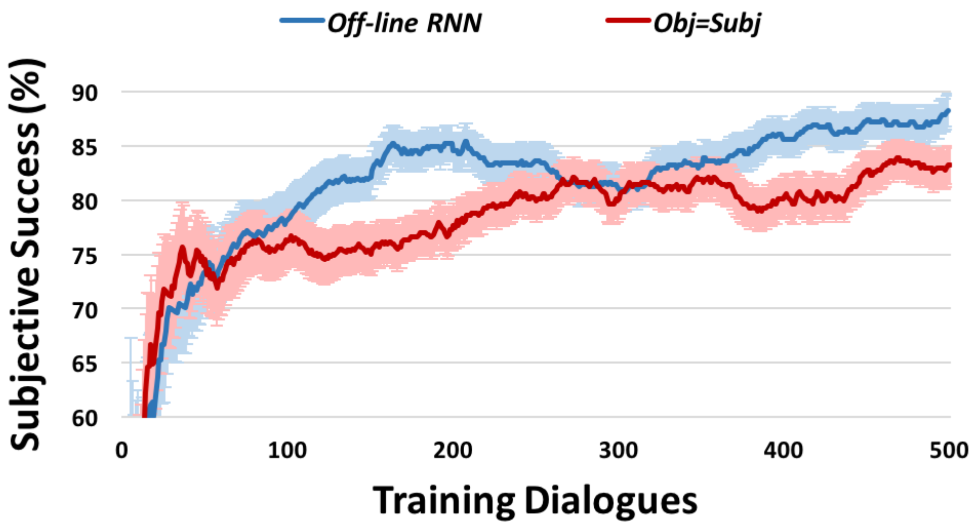


Figure 8: Learning curve for the reward plotted as a function of the number of training dialogues. The baseline system (red line) updates the policy only when the *Subj* and *Obj* measures agreed. The blue line shows training using the RNN dialogue success predictor. The lightly coloured bands denote 1 standard error.

6.2.2. Policy Learning with Human Users

Based on the above results, the binary RNN classification model with the feature vector $\mathcal{F}_{74}$ was selected for training dialogue policies on-line. Two systems were trained by users recruited via AMT. Firstly, an *Obj=Subj* system was built trained only with the dialogues whose user *Subj* rating matched the system *Obj* measurement. This of course requires knowledge of each task in order to compute the reward, and hence would not be viable for real users. However, it serves as a useful baseline.

Secondly, a system was trained using only the RNN to compute the reward signal. Note that a hand-crafted system is not used for comparison here since it does not scale to larger domains and is sensitive to speech recognition errors. Three policies were trained for each system, then averaged to statistically verify the performance. Figure 8 shows how the success rate improves as a function of the number of available training dialogues up to a maximum of 500 dialogues. For both plots, the moving average was calculated using a window of 150 dialogues and each result was the average of the three policies in order to reduce noise. It can be seen that the final performance of the RNN system resulted in a comparable policy to the baseline system. However, the baseline system actually required ~ 850 dialogues (due to discarding the cases where *Obj*$\neq$*Subj*). In contrast, the RNN system was more efficient and less costly since it used every dialogue.

The results indicate that the RNN dialogue success classifier was able to train a policy at least as well as the baseline system even though the baseline required prior knowledge of the users goal and selected only dialogues where the objective and subjective success estimates agreed.

6.3. On-line reward estimator

6.3.1. Supervised and Unsupervised Dialogue representations

The supervised RNN model described in Section 5.1.1 and the LSTM Encoder-Decoder model described in Section 5.1.2 were used to generate supervised and unsupervised representations $\mathbf{d_s}$ and $\mathbf{d_u}$ for each dialogue. The domain-independent feature vector $\mathcal{F}_{74}$ was used as the input of both embedding models and the target for the LSTM Encoder-Decoder model, where in the latter case the training objective was to minimise the MSE of the reconstruction loss.

For the supervised embedding, the size of hidden layer $\mathbf{d_s}$, was set to 32. In the unsupervised case, the sizes of $\overrightarrow{\mathbf{h_t}}$ and $\overleftarrow{\mathbf{h_t}}$ in the encoder and the hidden layer in the decoder were all 32, resulting in $dim(\mathbf{h_t}) = dim(\mathbf{d_u}) = 64$. SGD per dialogue was used to train each model. In order to prevent over-fitting, *early stopping* was applied based on the held-out validation set.

The data used to train the supervised dialogue embedding were the simulated dialogues `train-1K` and the validation set as specified in Section 6.2.1, and `testA` served as the test set. This dataset is denoted by `sim`. The embedding function was thus effectively the intermediate product created during the training of the RNN-based off-line reward model described in Section 6.2. For the unsupervised dialogue embedding, two datasets were investigated: firstly, the same dataset used to train the supervised embedding as mentioned above; and secondly, a corpus consisting of 8565, 1199 and 650 real user dialogues in the Cambridge restaurant domain was used for training, validation and testing respectively. This corpus was collected in previous user trials using paid subjects recruited via the AMT service. Thus this dataset is denoted by `amt`.

In order to visualise the effect of the embeddings, all of the test dialogues were transformed using the two embedding functions and the resulting representations $\mathbf{d}_\mathbf{s}^{\mathtt{sim}}$, $\mathbf{d}_\mathbf{u}^{\mathtt{sim}}$ and $\mathbf{d}_\mathbf{u}^{\mathtt{amt}}$ were plotted using t-SNE [70]. The results

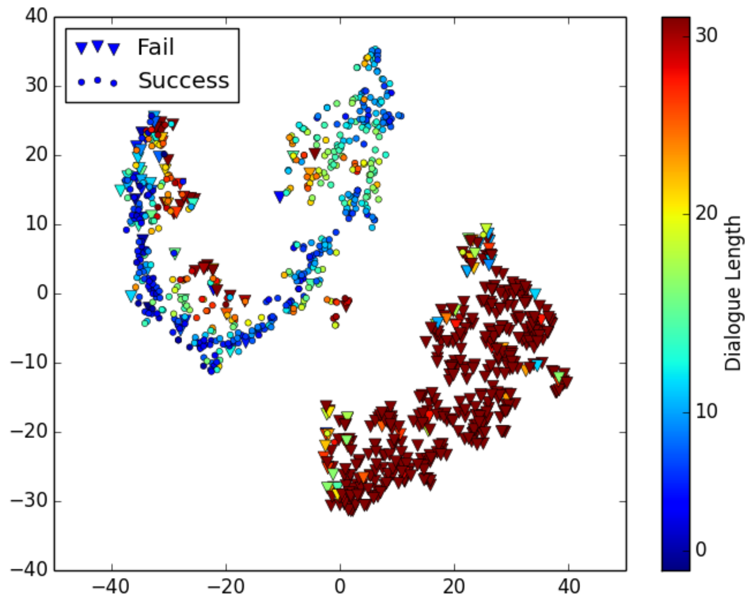
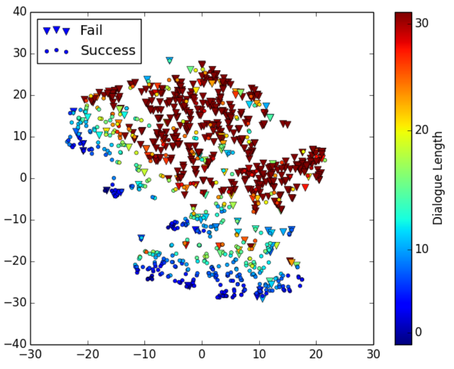


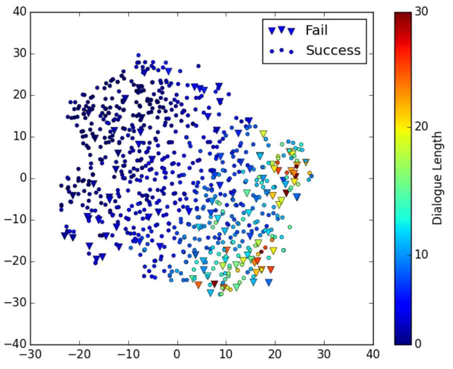
Figure 9: t-SNE visualisation on the supervised dialogue representations of the simulated data $\mathbf{d}_\mathbf{s}^{\texttt{sim}}$ in the Cambridge restaurant domain. Labels are the subjective ratings from the users and colours represent the total length of each dialogue.

are shown in Figures 9 and 10. For each dialogue sample, the shape indicates whether or not the dialogue was successful, and the colour indicates the length of the dialogue (maximum 30 turns). As can be seen in Figure 9, the supervised embedding function trained on simulated data clearly separates successful and failed dialogues into two categories. It is also clear that in simulation, the successful dialogues were mostly short and the failed ones were mostly aborted when the 30 turn limit was reached.

The patterns for the unsupervised embedding trained on both data are rather different, which are shown in sub-figures 10(a) & (b). In 10(a), the successful dialogues are distributed according to dialogue length from the bottom (shorter dialogues) to the top (longer dialogues). Compared to the patterns in Figure 9, the separation between successful and failed dialogues is less obvious, since the unsupervised embedding function was trained without success labels. Similar patterns can be found in sub-figure 10(b). These show that dialogue length was one of the most prominent features in the



(a) Unsupervised: $\mathbf{d}_\mathbf{u}^{\texttt{sim}}$



(b) Unsupervised: $\mathbf{d}_\mathbf{u}^{\texttt{amt}}$

Figure 10: t-SNE visualisation on (a) the unsupervised dialogue representations of the simulated data $\mathbf{d}_\mathbf{u}^{\texttt{sim}}$, and (b) the unsupervised dialogue representations of the real user data $\mathbf{d}_\mathbf{u}^{\texttt{amt}}$ in the Cambridge restaurant domain. Labels are the subjective ratings from the users and colours represent the total length of each dialogue.

unsupervised dialogue representations $\mathbf{d}_\mathbf{u}^{\texttt{sim}}$ and $\mathbf{d}_\mathbf{u}^{\texttt{amt}}$. Also for the dataset `amt`, the failed dialogues were distributed over all of the length range and the successful dialogues were on average shorter than 10 turns.

Investigating these two datasets `sim` and `amt`, it appears that whilst simulated and real data follow similar patterns in successful dialogues, typified by short conversations and clear information exchange between the user and the system, there is much more diversity in the real dialogues. This is especially the case for failed dialogues where real users halt the conversation as soon as they lose patience whereas simulated users persist until the maximum turn limit is reached. Note that for the simulated users, there is a patience setting which terminates the dialogue once the number of repetitive system responses reaches a certain threshold. However, real users demonstrate impatience in many different ways such as getting inconsistent system responses.

6.3.2. Dialogue Policy Learning

Given well-trained dialogue embedding functions, the proposed GP reward model operates on the embedded input space. The system was implemented using the GPy library [71]. Given the predictive success probability of each newly seen dialogue, the threshold λ for the uncertainty region was initially set to 1 to encourage label querying and annealed to 0.85 over the

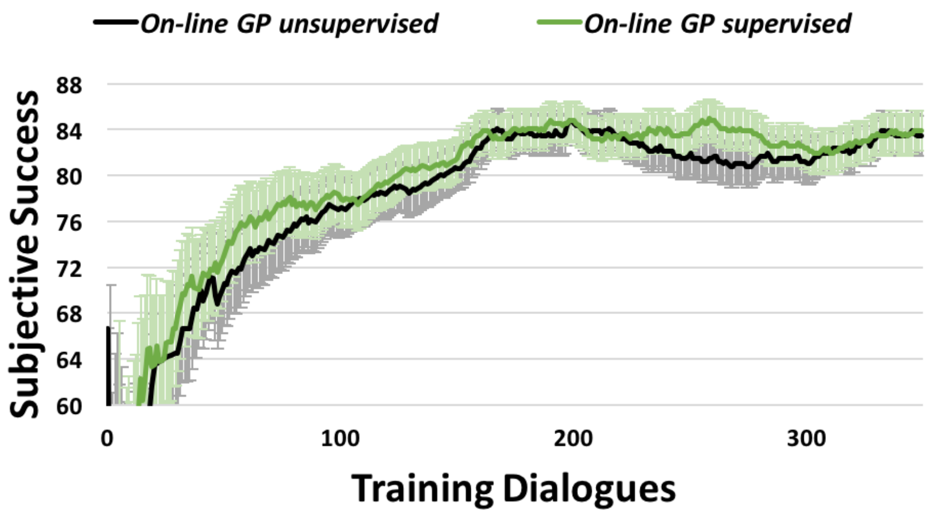


Figure 11: Learning curves showing subjective success as a function of the number of training dialogues used during on-line policy optimisation. The *on-line GP supervised*, *on-line GP unsupervised* and *Obj=Subj* systems are shown as green, black, and red lines. The light-coloured areas are one standard error intervals.

first 50 collected dialogues and then fixed at 0.85 thereafter. Initially, as each new dialogue was added to the training set, the hyper-parameters that define the kernel structure defined in Equation 8 were optimised to minimise the negative log marginal likelihood using conjugate gradient ascent [59]. To prevent *overfitting*, after the first 40 dialogues, these hyper-parameters were only re-optimised after every batch of 20 dialogues.

Two systems: *on-line GP supervised* and *on-line GP unsupervised* were trained with a total of around 350 dialogues on-line by users recruited via the AMT service. Figure 11 shows the on-line learning curve of subjective success during training. For each system, the moving average was calculated using a window of 150 dialogues. In each case, three distinct policies were trained and the results were averaged to reduce noise.

As can be seen, both systems had better than 80% subjective success rate after approximately 200 training dialogues. the supervised model performed slightly better during the learning process, and both converged to similar performance. This may be because the supervised embedding function was trained with knowledge of the objective success, and this is correlated to the subjective success ratings of real users. Nevertheless, after 300 dialogues

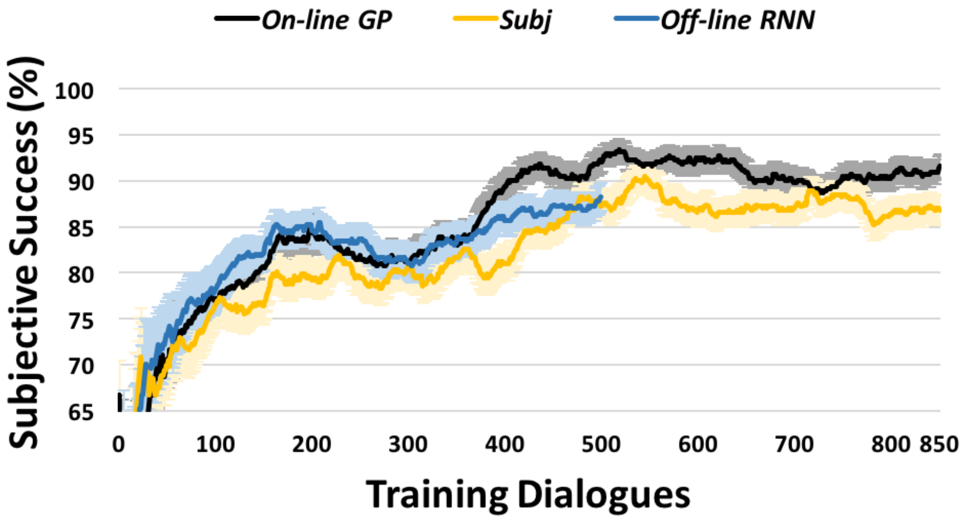


Figure 12: Learning curves showing subjective success as a function of the number of training dialogues used during on-line policy optimisation. The *on-line GP*, *Subj* and *off-line RNN* systems are shown as black, yellow, and blue lines. The light-coloured areas are one standard error intervals.

both embeddings achieve similar performance suggesting that the unsupervised embedding is perfectly adequate for the task, and it has the significant advantage that it can be trained on real data without annotations. In the following, the unsupervised method was therefore chosen for further investigation.

In addition to the above comparisons, two other systems were explored which used different methods to compute the reward:

- the *Subj* system which directly optimises the policy using only the user's subjective assessment of success whether accurate or not.
- the *off-line RNN* system that uses 1K simulated data and the corresponding *Obj* labels to train an RNN success estimator as in [21].

For the *Subj* system rating, in order to focus solely on the performance of the policy rather than other aspects of the system such as the fluency of the reply sentence, users were asked to rate dialogue success by answering the following question: *Did you find all the information you were looking for?*

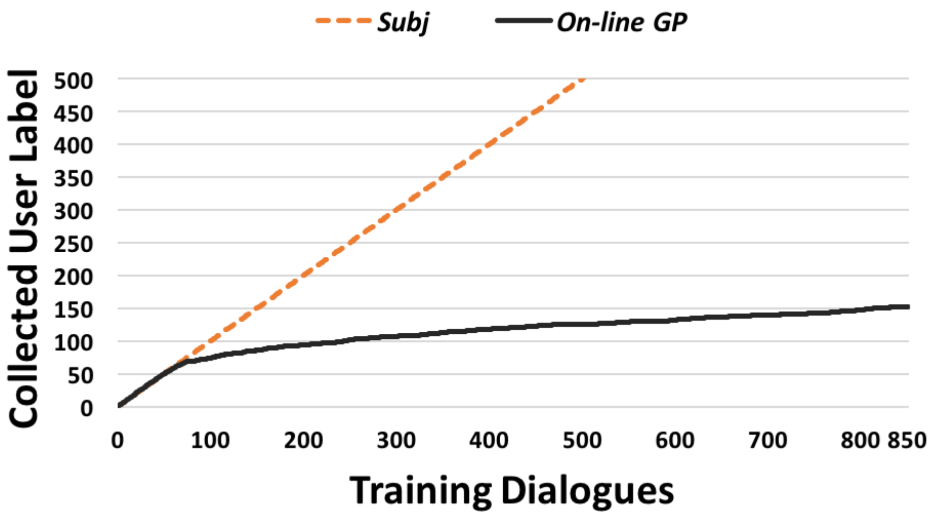


Figure 13: The number of times each system queries the user for feedback during on-line policy optimisation as a function of the number of training dialogues. The orange line represents the *Subj* system, and the black line stands for the *on-line GP* system.

Figure 12 shows the on-line learning curve of the subjective success rating with the moving average of 150 dialogues when training the three systems: *on-line GP* (with unsupervised embedding), *Subj*, and *off-line RNN*. In each case, three distinct policies were trained and the results were averaged to reduce noise. The systems were trained with a total of 500 dialogues on-line by users recruited via the AMT service.

We can clearly see that all three systems perform better than 85 % subjective success rate after approximately 500 training dialogues. To investigate learning behaviour over longer spans, training for the *on-line GP* and the *Subj* systems was extended to 850 dialogues. As can be seen, performance in both cases is broadly flat.

Similar to the conclusions drawn in [20], the *Subj* system appears to suffer from unreliable user feedback. As with the *Obj=Subj* system in Figure 8, this is partly due to users forgetting the full requirements of the task and in particular, they forget to ask for all required information. From Figure 12 it can be clearly seen that the *on-line GP* system consistently performed better than *Subj* system, presumably, because its noise model mitigates the effect of inconsistency in user feedback. Of course, unlike crowd-sourced subjects, real users might provide more consistent feedback, but nevertheless, some inconsistency is inevitable and the noise model offers the needed robustness.

The advantage of the *on-line GP* system in reducing the number of times that the system requests user feedback (i.e. the label cost) can be seen in Figure 13. The black curve shows the number of active learning queries triggered in the *on-line GP* system averaged across the three policies. This system required only 150 user feedback requests to train a robust reward model. On the other hand, the *Subj* system requires user feedback for every training dialogue as shown by the dashed orange line (of course, the same applies to the *Obj=Subj* system in the previous experiment).

Of course, the *off-line RNN* system required no user feedback at all when training the system on-line since it had the benefit of prior access to a user simulator. Its performance during training after the first 300 dialogues was, however, inferior to the *on-line GP* system.

6.3.3. Dialogue Policy Evaluation

In order to compare performance, the averaged results obtained between 400-500 training dialogues are shown in the first section of Table 1 along with one standard error. For the 400-500 interval, the *Subj*, *off-line RNN* and *on-line GP* systems achieved comparable results without statistical differences. The results of continuing training on the *Subj* and *on-line GP* systems from 500 to 850 training dialogues are also shown. As can be seen in the table, the *on-line GP* system was significantly better presumably because it is more robust to erroneous user feedback compared to the *Subj* system.

6.3.4. Reward Model Evaluation

The above results verify the effectiveness of the proposed reward model for policy learning. Here we investigate further the accuracy of the *on-line GP* model in predicting the subjective success rate. An evaluation of the *on-line GP* reward model between 1 and 850 training dialogues is presented in Table 2.

Since three reward models were learnt each with 850 dialogues, there were a total of 2550 training dialogues. Of these, the *on-line GP* model queried the user for feedback a total of 454 times, leaving 2096 dialogues for which learning relied on the reward model's prediction. Therefore, for a fair comparisons results shown in the table are averaged over 2096 dialogues.

As can be seen, there was a significant imbalance between successful and failed dialogues since the policy was improving along with the training dia-

Table 1: Subjective evaluation of the *off-line RNN*, *Subj* and *on-line GP* system during different stages of on-line policy learning. *Subjective*: user binary rating on dialogue success. Statistical significance was calculated using a two-tailed Students t-test with p-value of 0.05.

Dialogues	**Reward Model**	***Subjective* (%)**
400-500	*off-line RNN*	89.0 ± 1.8
	Subj	90.7 ± 1.7
	on-line GP	91.7 ± 1.6
500-850	*Subj*	87.1 ± 1.0
	on-line GP	$\mathbf{90.9 \pm 0.9}$*****

* $p < 0.05$

logues. This lowered the recall on failed dialogue prediction is expected since the model is biased toward successful dialogues. Overall, however, performance as judged by F-measure appears to be adequate.

Table 2: Statistical evaluation of the prediction of the *on-line GP* systems with respect to *Subj* rating.

Subj	**Precision**	**Recall**	**F-measure**	**Number**
Fail	1.00	0.52	0.68	204
Suc.	0.95	1.00	0.97	1892
Total	0.96	0.95	0.95	2096

6.4. Example Dialogues

The key benefits of the *on-line GP* reward model compared to other models are its robustness to noise and efficient use of user supervision. Since the four systems compared above differ only in the design of the reward model (learning objective), their on-line behaviours were broadly similar.

Two example dialogues between users and the *on-line GP* system are listed in Table 3 to illustrate how the system operates under different noise conditions. The user's subjective rating and the rating determined by the *on-line GP* reward model are also shown. The labels '*n*-th ASR' and '*n*-th SEM' indicate the n-th most likely hypotheses from the speech recogniser and the semantic decoder respectively.

Table 3: Example dialogues between on-line users and the proposed *on-line GP* system.

-Low noise condition- (User Subjective Evaluation: *Success*, **on-line GP Evaluation:** *Success***)**	
Machine	: Hello, welcome to the Cambridge dialogue system, what kind of food would you like?
Human	: [Top ASR] Yeah I want to find a restaurant that serves European food.
	[Top SEM] inform(food=european,type=restaurant)
Machine	: Hotel du Vin and Bistro is a nice place. It serves European food.
Human	: [Top ASR] Uh what is the address and phone number?
	[Top SEM] request(phone,addr)
Machine	: Hotel du Vin and Bistro is a nice place. Their address is 15-19 Trumpington Street and their phone is 01223 227330.
Human	: [Top ASR] Thank you goodbye.
	[Top SEM] bye()
Machine	: Thank you for using this system.
-High noise condition- (User Subjective Evaluation: *Success*, **on-line GP Evaluation:** *Success***)**	
Machine	: Hello, welcome to the Cambridge dialogue system, What kind of food would you like?
Human	: [Top ASR] And I want to find a expensive restaurant it should be in the south part of town.
	[Top SEM] inform(area=south,pricerange=expensive,type=restaurant)
Machine	: Gourmet burger kitchen is a nice place it is in the expensive price range.
Human	: [Top ASR] And do you have something else?
	[Top SEM] null()
	[2nd SEM] reqalts()
Machine	: Did you say you are looking for a venue in the area of south ?
Human	: [Top ASR] Yes.
	[Top SEM] affirm()
Machine	: Frankie and Benny's is a nice place, it is in the south part of town and it is in the expensive price range.
Human	: [Top ASR] Okay can I have the uh address phone number called for any.
	[Top SEM] request(addr)
	[2nd SEM] request(phone,addr)
Machine	: Frankie and Benny's is a nice place. Their address is Clifton Way and their phone number is 01223 412430.
Human	: [Top ASR] What is the pub called for free you any?
	...
	[5th ASR] What is the post code for free you any?
	[Top SEM] request(postcode)
Machine	: Frankie and Benny's is a nice place. Their postcode is C.B 1, 7 D.Y.
Human	: [Top ASR] Thank you goodbye.
	[Top SEM] bye()
Machine	: Thank you for using this system.

7. Conclusion

Computing a reliable reward function for training a dialogue policy on-line using reinforcement learning is difficult since the user's objective is usually unknown. Directly querying the user at the end of each dialogue is equally problematic since the resulting feedback is often inaccurate and repeated querying for feedback can be intrusive.

This paper has explored the design of a reward model which automatically estimates the success of a dialogue by monitoring the sequence of dialogue turns. Two approaches were investigated. Firstly, a recurrent neural network was utilised as a task success predictor which was pre-trained from off-line data. Secondly, an on-line learning framework was described whereby the dialogue policy is jointly trained alongside the reward function modelled as a Gaussian process with active learning. To provide a fixed size input space for the Gaussian process, variable length dialogue turn sequences are embedded using both a supervised recurrent neural network, and an unsupervised RNN-based encoder-decoder. Both perform similarly, but the latter has the advantage of requiring no task success annotation and can therefore be trained on unlabelled corpora.

This on-line reward model enables stable policy optimisation by robustly modelling the inherent noise in real user feedback and uses active learning to minimise the number of feedback requests to the user. The proposed on-line reward models achieved efficient policy learning and better performance compared to other state-of-the-art methods in the Cambridge restaurant domain. A key advantage of this Bayesian model is that its uncertainty estimate allows active learning and noise handling in a natural way. This mitigates the two key problems of requesting direct user feedback: it minimises the intrusive nature of asking for feedback, and it successfully filters inaccurate user responses.

Overall, the techniques developed in this paper enable for the first time viable approaches to on-line learning in deployed real-world dialogue systems. Dialogue policies can be learned from scratch without requiring the collection and annotation of bootstrap training data or the construction of a user simulator.

As with all of our previous work, the reward function studied here is focused primarily on task success and this may be too simplistic for many commercial applications. Further work may therefore be needed in conjunction with human interaction studies to identify and incorporate the further dimensions of dialogue quality needed to optimise user satisfaction. Nevertheless, task success is likely to remain the principal component of user satisfaction for the majority of applications, and hence the work reported here should be a major step forward in the practical deployment of systems which can learn and improve on-line without manual intervention.

8. Acknowledgement

P.-H. Su is supported by Cambridge Trust and the Ministry of Education, Taiwan. This research was partly funded by the EPSRC grant EP/M018946/1 Open Domain Statistical Spoken Dialogue Systems. The data used in the experiments is available at www.repository.cam.ac.uk/handle/1810/256020.

[1] O. Vinyals, Q. Le, A neural conversational model, arXiv preprint arXiv:1506.05869 (2015).

[2] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, J. Pineau, Hierarchical neural network generative models for movie dialogues, arXiv preprint arXiv:1507.04808 (2015).

[3] S. Young, M. Gašic, B. Thomson, J. Williams, Pomdp-based statistical spoken dialogue systems: a review, Proc of IEEE 99 (2013) 1–20.

[4] M. Henderson, M. Gašić, B. Thomson, P. Tsiakoulis, K. Yu, S. Young, Discriminative spoken language understanding using word confusion networks, IEEE SLT (2012).

[5] Y.-N. Chen, D. Hakkani-Tür, G. Tur, J. Gao, L. Deng, End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding, Proc of INTERSPEECH (2016).

[6] M. Henderson, B. Thomson, S. J. Young, Word-based Dialog State Tracking with Recurrent Neural Networks, Proc of SIGDIAL (2014).

[7] M. Gašić, S. Young, Gaussian processes for pomdp-based dialogue manager optimization, TASLP 22 (2014) 28–40.

[8] T.-H. Wen, M. Gašić, N. Mrkšić, P.-H. Su, D. Vandyke, S. Young, Semantically conditioned lstm-based natural language generation for spoken dialogue systems, Proc of EMNLP (2015).

[9] E. Levin, R. Pieraccini, A stochastic model of computer-human interaction for learning dialogue strategies., Proc of Eurospeech (1997).

[10] N. Roy, J. Pineau, S. Thrun, Spoken dialogue management using probabilistic reasoning, Proc of SIGDIAL (2000).

[11] J. D. Williams, S. Young, Partially observable Markov decision processes for spoken dialog systems, Computer Speech and Language 21 (2007) 393–422.

[12] F. Jurčíček, B. Thomson, S. Young, Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps, ACM TSLP (2011).

[13] J. Li, W. Monroe, A. Ritter, D. Jurafsky, Deep reinforcement learning for dialogue generation, Proc of EMNLP (2016).

[14] P.-H. Su, M. Gašić, N. Mrkšić, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, S. Young, Continuously learning neural dialogue management (2016).

[15] B. Dhingra, L. Li, X. Li, J. Gao, Y.-N. Chen, F. Ahmed, L. Deng, End-to-end reinforcement learning of dialogue agents for information access, arXiv preprint arXiv:1609.00777 (2016).

[16] P.-H. Su, P. Budzianowski, S. Ultes, M. Gasic, S. Young, Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management, Proc of SIGDIAL (2017).

[17] M. A. Walker, D. J. Litman, C. A. Kamm, A. Abella, PARADISE: A framework for evaluating spoken dialogue agents, Proc of EACL (1997).

[18] M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, S. J. Young, On-line policy optimisation of bayesian spoken dialogue systems via human interaction, Proc of ICASSP (2013).

[19] L. Zhao, G. Sukthankar, R. Sukthankar, Incremental relabeling for active learning with noisy crowdsourced annotations, Proc of PASSAT and Proc of SocialCom (2011).

[20] M. Gašić, F. Jurcicek, B. Thomson, K. Yu, S. Young, On-line policy optimisation of spoken dialogue systems via live interaction with human subjects, IEEE ASRU (2011).

[21] P.-H. Su, D. Vandyke, M. Gašić, D. Kim, N. Mrkšić, T.-H. Wen, S. Young, Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems., Proc of Interspeech (2015).

[22] D. Vandyke, P.-H. Su, M. Gašić, N. Mrkšić, T.-H. Wen, S. Young, Multi-domain dialogue success classifiers for policy training, IEEE ASRU (2015).

[23] J. Schatzmann, K. Weilhammer, M. Stuttle, S. Young, A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies, The knowledge engineering review (2006) 97–126.

[24] P.-H. Su, M. Gašić, N. Mrkšić, L. Rojas-Barahona, S. Ultes, D. Vandyke, T.-H. Wen, S. Young, On-line active reward learning for policy optimisation in spoken dialogue systems, Proc of ACL (2016).

[25] V. Rieser, O. Lemon, Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets, Computational Linguistics 37 (2011).

[26] L. Larsen, Issues in the evaluation of spoken dialogue systems using objective and subjective measures, IEEE ASRU (2003).

[27] Z. Yang, G. Levow, H. Meng, Predicting user satisfaction in spoken dialog system evaluation with collaborative filtering, IEEE Journal of Selected Topics in Signal Processing (2012) 971–981.

[28] L. El Asri, R. Laroche, O. Pietquin, Task completion transfer learning for reward inference, Proc of MLIS (2014).

[29] P.-H. Su, D. Vandyke, M. Gašić, N. Mrkšić, T.-H. Wen, S. Young, Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems, Proc of SIGDIAL (2015).

[30] S. Ultes, W. Minker, Quality-adaptive spoken dialogue initiative selection and implications on reward modelling, Proc of SIGDIAL (2015).

[31] P. Dai, C. H. Lin, D. S. Weld, et al., Pomdp-based control of workflows for crowdsourcing, Artificial Intelligence 202 (2013).

[32] C. H. Lin, D. S. Weld, et al., To re (label), or not to re (label), Second AAAI Conference on Human Computation and Crowdsourcing (2014).

[33] B. Settles, Active learning literature survey, Computer Sciences Technical Report 1648 (2010).

[34] C. Zhang, K. Chaudhuri, Active learning from weak and strong labelers, CoRR abs/1510.02847 (2015).

[35] E. Brochu, V. M. Cora, N. De Freitas, A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, arXiv preprint arXiv:1012.2599 (2010).

[36] C. Daniel, M. Viering, J. Metz, O. Kroemer, J. Peters, Active reward learning, Proc of RSS (2014).

[37] S. Russell, Learning agents for uncertain environments, Proc of COLT (1998).

[38] T. Paek, R. Pieraccini, Automating spoken dialogue management design using machine learning: An industry perspective, Speech communication (2008).

[39] A. Boularias, H. R. Chinaei, B. Chaib-draa, Learning the reward model of dialogue pomdps from data, NIPS Workshop on Machine Learning for Assistive Techniques (2010).

[40] L. M. Rojas Barahona, C. Cerisara, Bayesian Inverse Reinforcement Learning for Modeling Conversational Agents in a Virtual Environment., Conference on Intelligent Text Processing and Computational Linguistics (2014).

[41] W. Cheng, J. Fürnkranz, E. Hüllermeier, S.-H. Park, Preference-based policy iteration: Leveraging preference learning for reinforcement learning, Machine learning and knowledge discovery in databases (2011).

[42] H. Sugiyama, T. Meguro, Y. Minami, Preference-learning based inverse reinforcement learning for dialog control, Proc of Interspeech (2012).

[43] B. Thomson, S. Young, Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems., Computer Speech and Language 24 (2010) 562–588.

[44] S. Keizer, M. Gašić, F. Jurcicek, F. Mairesse, B. Thomson, K. Yu, S. Young, Proc of SIGDIAL (2010).

[45] Y. Engel, Algorithms and representations for reinforcement learning, PhD Thesis, 2005.

[46] T. Jebara, R. Kondor, A. Howard, Probability product kernels, J. Mach. Learn. Res. 5 (2004) 819–844.

[47] A. Karpathy, L. Fei-Fei, Deep visual-semantic alignments for generating image descriptions, CoRR abs/1412.2306 (2014).

[48] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, S. Khudanpur, Recurrent neural network based language model., Proc of Interspeech (2010).

[49] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, S. Khudanpur, Extensions of recurrent neural network language model, Proc of ICASSP (2011).

[50] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, et al., Using recurrent neural networks for slot filling in spoken language understanding, IEEE TASLP 23 (2015) 530–539.

[51] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, NIPS (2013).

[52] J. Turian, L. Ratinov, Y. Bengio, Word representations: a simple and general method for semi-supervised learning, Proc of ACL (2010).

[53] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, NIPS (2014).

[54] K. Cho, B. v. M. C. Gulcehre, D. Bahdanau, F. B. H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder–decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[55] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997).

[56] A. Graves, N. Jaitly, A.-r. Mohamed, Hybrid speech recognition with deep bidirectional lstm, IEEE ASRU (2013).

[57] J. P. Chiu, E. Nichols, Named entity recognition with bidirectional lstm-cnns, arXiv preprint arXiv:1511.08308 (2015).

[58] Z. Huang, W. Xu, K. Yu, Bidirectional lstm-crf models for sequence tagging, arXiv preprint arXiv:1508.01991 (2015).

[59] C. E. Rasmussen, C. Williams, Gaussian processes for machine learning (2006).

[60] I. n. Casanueva, T. Hain, H. Christensen, R. Marxer, P. Green, Knowledge transfer between speakers for personalised dialogue management, Proc of SIGDIAL (2015).

[61] D. Kim, C. Breslin, P. Tsiakoulis, M. Henderson, S. J. Young, Inverse reinforcement learning for micro-turn management., IEEE SLT (2014).

[62] L. Chen, P.-H. Su, M. Gašic, Hyper-parameter optimisation of gaussian process reinforcement learning for statistical dialogue management, Proc of SIGDIAL (2015).

[63] H. Nickisch, C. E. Rasmussen, Approximations for binary gaussian process classification, JMLR 9 (2008).

[64] A. Kapoor, K. Grauman, R. Urtasun, T. Darrell, Active learning with gaussian processes for object categorization, Proc of ICCV (2007).

[65] F. Jurčíček, S. Keizer, M. Gašić, F. Mairesse, B. Thomson, K. Yu, S. Young, Real user evaluation of spoken dialogue systems using Amazon Mechanical Turk, Proc of Interspeech (2011).

[66] M. Gašić, C. Breslin, M. Henderson, M. Szummer, B. Thomson, P. Tsiakoulis, S. Young, On-line policy optimisation of Bayesian Dialogue Systems by human interaction (2013).

[67] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, Y. Bengio, Theano: new features and speed improvements, Deep Learning and Unsupervised Feature Learning NIPS Workshop (2012).

[68] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, Theano: a CPU and GPU math expression compiler, Proceedings of the Python for Scientific Computing Conference (2010).

[69] J. Schatzmann, S. Young, The hidden agenda user simulation model, TASLP 17 (2009) 733–747.

[70] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, JMLR 9 (2008) 85.

[71] J. Hensman, N. Fusi, R. Andrade, N. Durrande, A. Saul, M. Zwiessele, N. D. Lawrence, GPy: A gaussian process framework in python, http://github.com/SheffieldML/GPy, 2012.