This paper tackles the problem of learning a dialog policy from example dialogs – for example, from Wizard-of-Oz style dialogs, where an expert (person) plays the role of the system. Learning in this setting is challenging because dialog is a temporal process in which actions affect the future course of the conversation – i.e., dialog requires planning. Past work solved this problem with either conventional supervised learning or reinforcement learning. Reinforcement learning provides a principled approach to planning, but requires more resources than a fixed corpus of examples, such as a dialog simulator or a reward function. Conventional supervised learning, by contrast, operates directly from example dialogs but does not take proper account of planning. We introduce a new algorithm called Temporal Supervised Learning which learns directly from example dialogs, while also taking proper account of planning. The key idea is to choose the next dialog action to maximize the expected discounted accuracy until the end of the dialog. On a dialog testbed in the calendar domain, in simulation, we show that a dialog manager trained with temporal supervised learning substantially outperforms a baseline trained using conventional supervised learning.

1. INTRODUCTION

In a spoken dialog system, the dialog policy is the component that examines the current state of the dialog, and decides what action to perform. In this paper, we are interested in the problem of learning a dialog policy from example dialogs. Providing example dialogs is often an easy method for a domain expert to express the desired behavior of a dialog policy – for example, an expert might be a developer of an existing smartphone application.

Past work has solved this problem using supervised learning, which predicts a dialog action given the current dialog state. The limitation of this approach is that the effects of an action on the future course of the dialog are not considered – i.e., proper multi-step planning is not done. As an illustration of this problem, consider the case where dialogs in a calendar domain are provided by multiple domain experts – e.g., multiple wizards in a wizard-of-oz setting. Consider a dialog state $s$ in which the time slot has been received from the user, but with low confidence. In $s$, suppose most experts choose to next request the date slot with action $a$, and a minority choose to confirm the date slot with action $a'$. After taking action $a$ in this state, the experts have a variety of error recovery mechanisms later in the dialog, and are therefore difficult to mimic, because of sparsity in the training data. On the other hand, after taking action $a'$ in this state, the experts are less stochastic and are easier to follow. This illustration shows that choosing the most common action in the current state does not necessarily maximize accuracy over the entire dialog: maximizing accuracy over the entire dialog requires balancing immediate accuracy and expected future accuracy by looking ahead.

A related method for policy learning is reinforcement learning. Reinforcement learning does do proper planning, but also requires the developer to design a reward function, which quantifies the goodness of a particular action taken in a particular dialog state. In practice this is difficult to do: domain experts often iteratively adjust the reward function until the resulting policy matches their expectations, suggesting experts may find it easier to give example dialogs rather than the reward function.

The paper introduces temporal supervised learning, an algorithm for learning a dialog policy that properly accounts for the effects of actions on the future, which learns only from example dialogs in a fixed dataset. The key idea is to choose the next dialog action to maximize the expected discounted accuracy until the end of the dialog, not merely the accuracy at the current time step. The algorithm handles input from multiple or noisy experts, and has only one free parameter which sets the trade-off between expected immediate accuracy and expected future accuracy.

In this paper, the next two sections formalize the problem and present related work; section 4 details the method; sections 5–6 cover the evaluation and results; and the last section concludes.

2. BACKGROUND

This paper is concerned with learning the dialog policy from example dialogs provided by one or more domain experts.
This problem is an instance of imitation learning, also known as learning from demonstration in the robotics literature [1].

Imitation learning (IL) for dialog systems can be characterized by the tuple $\langle S, A, T, D \rangle$, where $S$ is the set of dialog states, $A$ the set of dialog actions (assumed to be finite in this paper), $T$ the transition function $Pr(s'|s,a)$, and $D$ a set of dialogs. For example, one dialog state in $S$ could be the very start of the dialog, and another could be the situation where the system has heard the user wants to create a meeting today at a time not yet specified. Example actions include asking the user an open question like “How may I help you?” or a more directed question like “Ok, a meeting today, at what time?”.

Here, we assume that state transitions are Markovian. Every dialog $D \in D$ consists of an interleaving sequence of state–actions, $(s_1, a_1, s_2, a_2, \ldots, s_L)$, where $a_t$ is the action chosen by the expert in state $s_t$, the next-state $s_{t+1}$ are randomly drawn from the transition probability distribution $Pr(\cdot | s_t, a_t)$, $L$ is the dialog length. The goal of an imitation learning algorithm is to learn a policy $\pi$ that is similar to $\pi_e$, the (possibly stochastic) expert policy, from the dialogs.

Our solution to imitation learning, and discussion of existing work, will make use of reinforcement learning (RL) [2], so we also review it here. As above, we will assume state transitions are Markovian, which allows RL to be modeled as a Markov Decision Process (MDP) [3]. An MDP is characterized by a five-tuple $\langle S, A, T, R, \gamma \rangle$, where $S$, $A$, and $T$ are defined as in IL, $R$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. Given a policy $\pi$ that maps states to actions, its action-value function $Q^\pi(s, a)$ is the expected discounted total reward collected by taking action $a$ in state $s$ and following $\pi$ thereafter. Denote by $V^\pi(s) := Q^\pi(s, \pi(s))$ the state-value function. In RL, the transition and reward functions are typically unknown, and the goal of an RL algorithm is to learn a policy that maximizes its value function [2] from past data. We denote an optimal policy and the optimal value functions by $\pi^*$, $V^*$, and $Q^*$, respectively.

3. RELATED WORK

Broadly speaking, most existing work on imitation learning for dialog systems has fallen into two categories: supervised learning and inverse reinforcement learning. First, standard supervised learning has been applied to learn a mapping from dialog state to action [4, 5, 6, 7]. Here, the learning algorithm is not explicitly aware of the effects of its choices on the future, i.e. it assumes state–action pairs are IID along the dialog, which does not hold, as in the example in Section 1.

Outside of dialog policy learning, algorithms from the machine-learning literature have been developed which extend supervised learning to account for this problem [8, 9]. However, they have requirements that go beyond a fixed corpus – for example, asking the expert for additional labels, or domain-specific heuristics.

A second approach to imitation learning for dialog systems has been to infer a reward function from dialog data using inverse reinforcement learning (IRL) [10, 11, 12]. IRL takes dialog data as input, and infers the reward function for which the policy followed in the data is optimal. Once that reward function has been learned, normal RL can be applied. Although IRL-based imitation learning does do proper planning, it assumes access to more than just example dialogs. In particular, past work has assumed that the dialog policy learner can explore new state–action pairs not observed in the dialog data. In real settings, this can be unrealistic, since commercial service providers are reluctant to experiment on customers, risking dissatisfaction.

RL has been applied to dialog policy learning extensively, including algorithms designed to learn from a fixed corpus [13]. However, RL is solving a different problem: imitation learning seeks to imitate an observed policy, and RL chooses actions to maximize rewards. As explained already, it is often nontrivial to design a reward function that yields a policy desired by a system designer.

In sum, there is no existing work on dialog policy learning – or, to our knowledge, imitation learning/learning from demonstration in any domain – that takes proper account of planning but requires only example dialogs. This is the problem this work addresses.

4. METHOD

Our goal is to learn a mapping $\pi$ from dialog states to dialog actions, given a corpus of dialogs $D$ from one or more experts. We assume that an expert may be deterministic or stochastic.

As mentioned above, casting the learning problem naively as supervised learning (SL) can lead to poor performance. The key problem is that – as a consequence of the IID assumption – SL does not consider the effects of its output on the future. The intuition of our approach is to incorporate temporal information into the training of the classifier. Specifically, we learn a multi-class classifier that chooses an action given the current state in order to minimize the expected discounted error for the entire dialog.

We start by defining $P_e(a|s)$, the probability of observing $a$ in expert-generated dialogs given state $s$, without regard to the future. Intuitively, we want to minimize misclassification rate not just in the current state, but also in all future reachable states. This rate in state $s$ is $1 - P_e(a|s)$, by definition. To incorporate temporal information, we will employ the machinery of RL, and define an expert-induced reward function as the negation of the misclassification rate:

$$R_e(s,a) := P_e(a|s) - 1, \quad (1)$$

The RL discount $\gamma \in [0, 1]$ specifies how much weight to place on misclassification at the current timestep versus the
future. If $\gamma = 0$, this policy reduces to a myopic policy that
is indifferent to the future, and is thus equivalent to a typical
supervised learning approach. As $\gamma$ increases, more weight is
placed on accuracy in the future.

With the addition of $R_e$ and $\gamma$, RL can now be applied
to dialogs to produce a policy, using any batch reinforcement
learning algorithms [14], such as experience replay [15],
least-squares policy iteration [16], and fitted Q-iteration [17].
Note that TSL is agnostic to the type of action, such as
requesting/confirming/presenting information, or querying a
database. From the standpoint of TSL, the only relevant fac-
tor is whether experts take this action in the current state, and
what successor states result.

To implement the method, two practical issues arise. The first is exploration in the example dialogs. Batch RL algo-
rithms assume that the training dialogs include sufficient
exploration; if action $a$ is never attempted in state $s$, the pol-
icy may be poorly learned in those states. In our experiments,
we collected dialogs both with and without exploration. As
shown later, TSL outperforms SL in both cases, and has
strong performance even without exploration.

The second issue is how to estimate the expert-induced re-
ward function $R_e(s, a)$. We propose a model-based approach:
from the dialogs, we learn a multi-class classifier which out-
puts a distribution $P_t(a|s)$ over all $a$ for a given $s$ as an esti-
mate of the expert policy. In our work, we use a multinomial
logistic regression model, since its output is well-calibrated,
but we note that calibration techniques can be used to convert
any classifier’s predictions to label probabilities.

This estimated multinomial distribution naturally encodes
some of the key properties required for IL. Consider a state $s$
which is visited often. In $s$, if experts all agree on an action $a^*$,
$R_e(s, a)$ will be nearly 0 for $a = a^*$ and nearly $-1$ for $a \neq
a^*$. If experts disagree on which actions to take in $s$, $P_t(a|s)$
will spread mass out among those actions – for example, for
2 equi-probable actions in $s$, $P_t(a|s) = 0.5$ and $R_e(s, a) =
-0.5$ for both $a$. Therefore, among frequently-visited states,
TSL will prefer states where experts agree more, since that
leads to higher expected rewards. Next, consider a state $s$
where there is little or no data. Here, $P_t(a|s)$ will spread mass
out among many actions (due to, say, regularization), so all
actions will yield low reward. TSL will avoid visiting these
states. Moreover, this reward also encourages the agent to fin-
ish episodic tasks early, since all rewards are non-negative.

4.1. Theoretical justifications

This section formalizes two intuitions that our method rests
on. Let $M = \langle S, A, T, R, \gamma \rangle$ be the original decision mak-
ing problem where $R$ is the unknown target reward func-
tion that is hard to specify. With the optimal value function
$Q^*$, the advantage function [18] is defined by: $A(s, a) :=
\max_b Q^*(s, b) - Q^*(s, a)$. A policy $\pi$ is $\varepsilon$-optimal in MDP
$M$, if $V^*(s) - V^\pi(s) \leq \varepsilon$ for any $s \in S$. With $R$ replaced
by the expert-induced reward function (Equation 1) in $M$, we
have a new MDP $M_e$, for which quantities like $V^\pi, Q^*,$ and
$A_e$ can be defined similarly.

The first observation is that a difference in distributions in
states encountered at training and testing time can cause a
supervised-learning algorithm to produce a policy with poor
test performance.

Proposition 1 Let $A$ be a supervised-learning algorithm de-
digned to minimize classification error for states encountered
in example dialogs generated by expert $E$. Assume the (pos-
sibly stochastic) expert policy, $\pi_e$, is $\varepsilon_1$-optimal in $M$, and
$A$ returns a policy $\pi_{SL}$ whose classification error is $\varepsilon_2$ in di-
als generated by $E$, for some positive $\varepsilon_2$. Then, no matter
how small $\varepsilon_1$ and $\varepsilon_2$ are, it is possible to construct an MDP
$M$ and expert $E$ such that $\pi_{SL}$ achieves the lowest possible
rewards among all possible policies, when it is run in $M$.

The proposition can be proved easily with examples, such as
the example in Section 1. It should be noted that the classifi-
cation error in the proposition above refers to test error, as
opposed to training error evaluated on training dialogs.

The second observation is that, minimizing the cumula-
tive classification error is fundamentally related to solving the
original problem: if the expert policy is near-optimal with re-
spect to the unknown reward function, then TSL indeed gives
a policy with a certain performance guarantee.

Proposition 2 Assume the (possibly stochastic) expert policy,
$\pi_e$, is $\varepsilon_1$-optimal in $M$, and a learned policy, $\hat{\pi}$, is $\varepsilon_2$-optimal
in $M_e$. Then, $\hat{\pi}$ is $\varepsilon$-optimal in $M$, where $\varepsilon = \varepsilon_1 + \varepsilon_2 A_{max}$
and $A_{max} = \max_{s,a} A(s, a)$.

Proof (sketch) Let $s$ be an arbitrary state, and $V^\pi(s)$ its value de-
finied using the target reward $R$. Now consider dialogs generated by
$\pi$ from $s$. The near-optimality of $\pi$ (in $M_e$) implies that, the expected
discounted number of times $\hat{\pi}$ deviates from $\pi_e$ in these dialogs is at
most $\varepsilon_2$. Each such deviation contributes to the gap $V^\pi(e) - V^\pi(s)$
by at most $A_{max}$. Combined with near-optimality condition of $\pi_e$ in
$M$, it follows that $V^\pi(s) - V^\pi(s)$ is at most $\varepsilon$. □

The two propositions above together show that, while
(non-temporal) SL may lead to a policy with poor test-time
performance, the objective in TSL is sound in principle: op-
timizing against the expert reward $R_e$ does indeed result in
a policy with controlled performance in the original decision
making problem, even if the reward function is not specified.

5. EXPERIMENTAL DESIGN

This experiment draws on an end-to-end simulation of a spo-
en dialog system which enables a user to query a calendar, and
to create, delete, and update appointments. An appointment
consists of 4 values: a time, date, location, and person the
meeting is with. The system follows the same architecture
as machine learning-based dialog systems common in the research literature [19, 20].

The mechanics of action selection are shown in Figure 1. In the left column, the user says a command to the system such as “Good morning. Create a new meeting at 3 after lunch”. A simulated speech recognizer and natural language understanding unit convert this into a semantic parse, shown in the second column. Note that this recognizer and parser may make mistakes – in this example, “3 after lunch” is recognized as “3 AM” because of the keyword “morning” in the user input. Speech recognition errors are also simulated.

The semantic parse is passed to a dialog state tracker, which maintains a persistent dialog state, shown in the third column. This process is based on simple hand-written update rules that accumulate information observed over the course of the dialog. The state tracker then outputs one or more action candidates, shown in the fourth column. Each action candidate is described in terms of the semantic content it contains, and can be rendered into natural language. In Figure 1, three candidates are shown; in practice, there can be hundreds or thousands, depending on the dialog state.

For each candidate, a vector of 1059 features are extracted (fifth column). Each feature is either binary or real-valued. The features do not contain actual semantic values (like “3 AM”), but rather indicate which value types are present, what confidence they have been recognized with, etc. Finally, the agent’s policy is applied, producing a state–action score for each candidate, and the action with the maximum score is output to the user. The whole cycle then repeats.

To create dialogs from which to learn, a bank of 15 expert policies were hand-created. These expert policies were deterministic functions of the persistent dialog state, but because the state–action features used by TSL were simpler, they were not deterministic functions of the features – i.e., just as with example dialogs provided by real experts, the simulated experts’ policies are partially observable w.r.t. the features. To generate a training dialog, an expert is sampled uniformly, and then that expert is used throughout the dialog. Although there are limitations to dialog simulation, simulation does allow us to evaluate whether an action taken following a learned policy would have been taken by a (simulated) expert – a measurement that would be very costly with real experts.

To enable an expert or learned agent to interact with the environment, a simulated user was created. The user has a persistent goal throughout the conversation, and varied but coherent actions. In addition, speech recognition and understanding errors were simulated, so that about half the parsed user inputs contained at least one semantic error. The dialog continues until the system takes a transactional action, such as creating or deleting an appointment, after which the user ends the dialog. If the dialog goes on for more than 20 exchanges, the simulated user gives up and abandons the conversation.

Given dialogs collected as described above, we used multinomial logistic regression to obtain an estimate \( \hat{R}_e \) of the expert-induced reward function. A SL policy would simply return a greedy action with respect to this estimate: for every \( s \in S \), \( \pi_{\text{SL}}(s) = \arg \max_{a \in A} \hat{R}_e(s, a) \). In contrast, the TSL agent uses Q-learning with experience replay and linear function approximation to (approximately) optimize total expert-induced rewards, and returns a non-myopic policy, denoted \( \pi_{\text{TSL}} \). All meta-parameters in multinomial logistic regression and Q-learning were tuned on a separate set of dialogs. With tuned meta-parameters, both learners were run

2Note that these hand-coded policies are used only to mimic experts. All learning algorithms below only have access to example dialogs generated by these policies, but not the policies themselves.

Fig. 1: Architecture of the end-to-end spoken dialog system. Components are described in the text.
6. RESULTS

Figures 2a-2d show average accuracies and task completion rates for policies learned on dialogs both with and without exploration. In both cases, the accuracies in Figures 2a and 2c are quite low, partly because the large number of candidate actions that can be at the level of thousands, and partly because the data is quite noisy: two randomly sampled experts agree on the optimal action in only 22% of turns. TSL accuracies are lower than SL accuracies, likely because TSL uses simple linear models, and SL uses the logistic regression that is more suitable to modeling probabilistic binary outputs.

Figures 2b and 2d show task completion rates. Here, TSL yields substantial gains over myopic SL – statistically significant at the 95% level – and is maximal when using a moderate amount of look-ahead: $\gamma = 0.5$. This result shows the benefit of incorporating look-ahead into imitation learning for dialog policy learning.

7. CONCLUSIONS

In this paper, we have shown that in imitation learning, myopically following the expert can lead to poorly learned poli-
cies. In contrast to previous iterative methods that require exploration in the environment and on-going access to an expert, we have introduced a batch policy learning algorithm that only needs a fixed set of expert dialogs. The experts can be heterogeneous and/or noisy. By maximizing the cumulative accuracy with respect to the expert(s) over a dialog, we leverage temporal information to avoid regions where experts show non-deterministic behavior or where training data is sparse. On a realistic simulated spoken dialog system, our method exceeds direct supervised learning in task completion rate.

There are several directions for future research. First, we would like to explore other methods for estimating expert-induced reward values, for example, data smoothing methods such as kernel density estimation, or confidence estimation models using application-specific confidence features. Second, we can explicitly tackle the problem of state distribution mismatch during training and testing by reweighing training examples – actions more likely to be taken at test time than at training time are weighted higher. Third, our approach can be easily extended to settings where exploration with expert input is affordable and expects improved performance. Finally, we plan to evaluate with a real end-to-end dialog system.

8. ACKNOWLEDGEMENTS
Thanks to Dan Bohus for making his maxent software available to us in the experiments.

9. REFERENCES