MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Log-linear dialog manager

Tang, H.; Watanabe, S.; Marks, T.K.; Hershey, J.R.

TR2014-024 May 2014

Abstract

We design a log-linear probabilistic model for solving the dialog management task. In both planning and learning we optimize the same objective function: the expected reward. Rather than performing full policy optimization, we perform on-line estimation of the optimal action as a belief-propagation inference step. We employ context-free grammars to describe our variable spaces, which enables us to define rich features. To scale our approach to large variable spaces, we use particle belief propagation. Experiments show that the model is able to choose system actions that yield a high expected reward, outperforming its POMDP-like log-linear counterpart and a hand-crafted rule-based system.

ICASSP 2014

LOG-LINEAR DIALOG MANAGER

Hao Tang[*]

Toyota Technological Institute at Chicago
6045 S. Kenwood Ave., Chicago, IL

Shinji Watanabe, Tim K. Marks, and John R. Hershey

Mitsubishi Electric Research Laboratories (MERL)
201 Broadway, Cambridge, MA

ABSTRACT

We design a log-linear probabilistic model for solving the dialog management task. In both planning and learning we optimize the same objective function: the expected reward. Rather than performing full policy optimization, we perform on-line estimation of the optimal action as a belief-propagation inference step. We employ context-free grammars to describe our variable spaces, which enables us to define rich features. To scale our approach to large variable spaces, we use particle belief propagation. Experiments show that the model is able to choose system actions that yield a high expected reward, outperforming its POMDP-like log-linear counterpart and a hand-crafted rule-based system.

Index Terms— Log-linear Model, POMDP, Dialog Manager

1. INTRODUCTION

A current trend in dialog manager research is to model dialog sessions using partially observable Markov decision processes (POMDPs) [1]. After the seminal work of [2] and more than a decade of subsequent research, statistical dialog systems have become a new standard for advanced dialog systems [3, 4, 5]. For recent developments in spoken dialog systems, see [6, 1] and the citations therein.

In a POMDP dialog system, the dialog is represented by means of a set of random variables at each turn of the dialog: an observed variable representing what the user has said, a hidden state variable representing the progress of the dialog so far, and a system action that has to be selected. The POMDP model defines two probabilistic dependencies: the conditional probability of the current state given the previous state and system action, and the conditional probability of the observation given the current state and previous system action.

A *reward function* specifies, for each turn, a fitness criterion as a function of the state and chosen action for that turn. Given a reward function, it is possible to determine a *policy* that provides the optimal system action given what is known about the state distribution at the current time. This policy can then be used to generate system actions in the course of a dialog. Selecting system actions is order to maximize reward is called *planning*.

To have a working system, one also needs to estimate the model parameters that define probabilities in the POMDP. This estimation is called *learning*. The parameters are typically estimated using a maximum likelihood criterion, rather than using the reward function. For example, a maximum likelihood dynamic Bayesian network (DBN) is used in [7]. A major problem with these approaches is that planning and learning are optimized separately using different criteria. In addition, planning and learning are notoriously difficult optimization problems [8]. This motivates us to design a simple coherent system that has the advantage of a consistent optimization criterion and at the same time is more efficient to optimize.

Our approach is to model the dialog system using a log-linear probability distribution. We call this a *log-linear dialog manager*. Log-linear distributions have been increasingly used to model sequences since the introduction of conditional random fields [9]. Although log-linear models in general cannot represent all distribution families, their flexible use of feature functions enables them to express a wide family of probabilistic models. In addition, since our model is a Markov chain, we can exploit efficient algorithms for optimization. In particular, we are interested in optimizing the sum of rewards along the time axis. Similar optimizations are well known for other problems [10, 11].

To represent the space of possible states, user actions, and system actions, we use *context-free grammars* (CFGs), each of which is based on a graph of semantic representations related to the domain of the dialog system. Instead of being simple multinomials, the random variables take values in the space of parse trees generated by the CFGs. This provides a rich structure that allows us to extract a wide range of features, and is reminiscent of [12, 3], which use graphs or rules to partition the space. Because of the flexible use of features inherent in log-linear models, we can design features that make our dialog system behave like a deterministic rule-based dialog system as a special case. This is done by implementing the rules of the deterministic dialog system as indicator-function features, and initializing the parameters such that the log-linear probability distributions correspond to these rules. As we obtain more data, the learning algorithm can provide a smooth transition from a rule-based system to a statistical data-driven system, which is a desirable feature in realword scenarios [13].

A common problem in dialog management is that inference becomes intractable in variable spaces large enough to handle real problems. This problem has been addressed in a variety of ways [14, 15, 16, 17]. With our model the optimization can be solved using belief propagation; we employ particle belief propagation [18] to scale the algorithm to large variable spaces.

2. MODEL DEFINITION

Our probabilistic model has four variables at each time step. Two are observable variables: the system action a_t and the observation o_t. The other two are latent variables which must be inferred: the user action u_t and the state s_t. Roughly speaking, each step of the dialog proceeds as follows. Based on all of the system actions and observations up to time $t-1$, the system prompts the user with query a_{t-1}. The user's response to that query is represented by o_t (in our system, o_t is a sequence of words uttered by the user). The meaning of that response is represented by the user action, u_t, which may be inferred from the observation. The new state, s_t, may be inferred based on the

[*]This work was performed while Hao Tang was working at Mitsubishi Electric Research Laboratories (MERL).

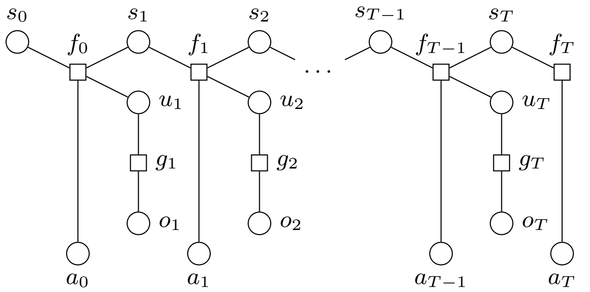

Fig. 1. The factor graph representation for the distribution in (1).

aforementioned system action a_{t-1} and user action u_t, as well the previous state s_{t-1}. In our system, the state s_t represents the user's intention, although in general it could also include additional contextual information. Using subscripted colons to denote sequences $\big($e.g., $s_{0:T} \equiv \{s_0, s_1, \ldots, s_T\}\big)$, an entire dialog session of length T is represented by four variable sequences: $s_{0:T}, a_{0:T}, o_{1:T}, u_{1:T}$.

Our model for a dialog session is represented by the factor graph in Fig. 1, which for our log-linear model corresponds to the following joint probability distribution over the variables:

$$p(s_{0:T}, a_{0:T}, u_{1:T}, o_{1:T}) = \tag{1}$$
$$\frac{1}{Z_\theta} \exp\left[\sum_{t=0}^{T} \boldsymbol{\theta}_f^\top \boldsymbol{\phi}_f(s_t, a_t, s_{t+1}, u_{t+1}) + \sum_{t=1}^{T} \boldsymbol{\theta}_g^\top \boldsymbol{\phi}_g(u_t, o_t)\right],$$

where Z_θ is a normalizing constant, $\boldsymbol{\phi}_f$ and $\boldsymbol{\phi}_g$ are vectors of feature functions, and $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$, respectively, are vectors of the corresponding model parameters. (Note that at time $t = T$, s_{t+1} and u_{t+1} are undefined, so as shown in factor f_T of the factor graph in Fig. 1, at time $t = T$ we define $\boldsymbol{\phi}_f$ as a function of only its first two inputs.) To simplify notation, we also define the following vectors:

$$\boldsymbol{\theta} \equiv \begin{bmatrix} \boldsymbol{\theta}_f \\ \boldsymbol{\theta}_g \end{bmatrix}, \qquad \boldsymbol{\phi}(t) \equiv \begin{bmatrix} \boldsymbol{\phi}_f(s_t, a_t, s_{t+1}, u_{t+1}) \\ \boldsymbol{\phi}_g(u_t, o_t) \end{bmatrix}. \tag{2}$$

These enable us to rewrite (1) more succinctly as

$$p(s_{0:T}, a_{0:T}, u_{1:T}, o_{1:T}) = \frac{1}{Z_\theta} \exp\left[\sum_{t=0}^{T} \boldsymbol{\theta}^\top \boldsymbol{\phi}(t)\right], \tag{3}$$

where

$$Z_\theta = \sum_{\substack{s_{0:T}, a_{0:T}, \\ u_{1:T}, o_{1:T}}} \exp\left[\sum_{t=0}^{T} \boldsymbol{\theta}^\top \boldsymbol{\phi}(t)\right]. \tag{4}$$

2.1. Variable Spaces

We let S, U, A, and O represent the variable spaces (the set of all possible values) for the variables s_t, u_t, a_t, and o_t, respectively. Each observation $o \in O$ can in theory be anything from waveforms or acoustic features to recognized texts. In this work, we let $o \in O$ represent the input word sequence, and we define the variable space O as the set of all sequences of words in a vocabulary set $\mathcal{V}$.

We define each of the variable spaces S, U, and A using a context-free grammar (CFG), which consists of a set of production rules. Each variable space is defined as the set of all possible parse trees that can be generated by its CFG. Fig. 3 shows some of the production rules in the CFG that defines the variable space S. Each parse tree in S is a possible value of the state s_t. Fig. 2 shows one possible value for state s_t (one parse tree in S), which was generated using the production rules shown in boldface in Fig. 3.

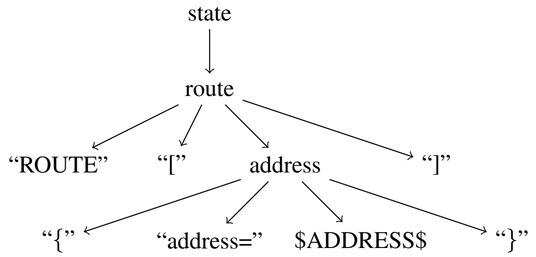

Fig. 2. An example of a parse tree for the state ROUTE[{address=$ADDRESS$}]. Terminals are enclosed in quotation marks, while nonterminals are not. The variable $ADDRESS$ can either be further extended with other production rules or remain as a free variable.

state → **route** | search | time_to_dest

route → **“ROUTE” “[” address “]”**

address → **“{” “address=” $ADDRESS$ “}”**

| “{” “address=” “NULL” “}”

search → “SEARCH” “[” keyword “]”

keyword → “{” “keyword=” $KEYWORD$ “}”

| “{” “keyword=” “NULL” “}”

time_to_dest → “TIME_TO_DEST” “[” route “]”

Fig. 3. A few of the production rules in the CFG that defines the variable space S. The production rules that are active in the parse tree of Fig. 2 are highlighted in bold.

2.2. Features

As can be seen in the factor graph in Fig. 1 and in (1), there are two types of factors in our model. The first, denoted f, models statistical dependencies between the previous and current state, the system action, and the user action. The second, denoted g, models dependencies between observed word sequences and their semantic interpretations. For the variables whose spaces are defined using CFGs, we treat each variable value (each parse tree) as a set of active production rules. For example, the production rules that are active in the parse tree of Fig. 2 are shown in boldface in Fig. 3.

Suppose G_S, G_U, and G_A are the set of production rules in the CFGs that define the variable spaces for S (states), U (user actions), and A (system actions), respectively. For factor g, we associate each production rule in a user action with a language model for the associated word sequences. Specifically, given a user action u_t and observation o_t, we have features of the form $\mathbb{1}_{k \in u_t, w_{i-1}w_i \in o_t}$, which denotes an indicator function that equals 1 if and only if a particular production rule $k \in G_U$ is active in the parse tree of user action u_t and a particular bigram $w_{i-1}w_i$ is present in the word sequence of observation o_t. The language model for a production rule that appears close to the root of the tree models a general class of utterance, whereas production rules that appear close to the leaves of the tree must be more specialized. For factor f, we can consider production rules that co-occur. For example, the feature $\mathbb{1}_{k \in s_{t-1}, k' \in s_t}$, which concerns two particular production rules $k, k' \in G_S$, equals 1 if and only if k is active in state s_{t-1} and k' is active in state s_t. Another type of feature typically seen in conventional dialog systems is $\mathbb{1}_{k \in s_{t-1}, k' \in s_t, j \in a_{t-1}}$, which additionally requires that production rule $j \in G_A$ is active in system action a_{t-1}. This feature indicates that a particular system action tends to induce a particular state transition.

3. PLANNING AND LEARNING

The two basic problems a dialog manager needs to solve are planning and learning. We assume there is a reward function $r : S \times A \to \mathbb{R}^+$ that assesses our model. This section explains how we do planning and learning in terms of the reward function.

3.1. Planning

Planning at time τ is the problem of finding the best system action a_τ, given the history of all previous system actions $a_{0:\tau-1}$ and observations $o_{1:\tau}$. Suppose the dialog has length T. We define the planning problem as finding a_τ to maximize the expected reward

$$\mathop{\mathbb{E}}_{\substack{s_{0:T}, a_{\tau+1:T}, \\ u_{1:T}, o_{\tau+1:T}}} \left[\frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t) \,\middle|\, a_{0:\tau-1}, o_{1:\tau} \right]. \qquad (5)$$

The expectation is taken over all variables not given: all states, all user actions, and all future system actions and observations.

The above objective could be optimized exactly by hypothesizing each action a_τ, computing the expected reward given that action using the sum-product algorithm, and selecting the action that maximized expected reward. However, for ease of implementation and speed, we instead optimize the objective's variational lower bound,

$$\mathop{\mathbb{E}}_{\substack{s_{0:T}, a_{\tau+1:T}, \\ u_{1:T}, o_{\tau+1:T}}} \left[\prod_{t=0}^{T} \left(\frac{r(s_t, a_t)}{\gamma_t (T+1)} \right)^{\gamma_t} \,\middle|\, a_{0:\tau-1}, o_{1:\tau} \right], \qquad (6)$$

obtained from Jensen's inequality, where the γ_t are variational parameters such that $\sum_t \gamma_t = 1$. Although the γ_t could be optimized, we take them to be uniform, setting $\gamma_t = 1/(T+1)$, to further simplify the computation.

This product form has the nice property that the reward factorizes with time. In other words, (6) can be expanded to

$$\frac{1}{Z'} \exp \left[\sum_{t=0}^{T} \left[\boldsymbol{\theta}^\top \boldsymbol{\phi}(t) + \gamma_t \log \left(\frac{r(s_t, a_t)}{\gamma_t (T+1)} \right) \right] \right], \qquad (7)$$

where Z' is the partition function of p with $a_{0:\tau-1}$, $o_{1:\tau}$ given. Now finding the best a_τ involves just running a standard sum-product algorithm on the graphical model with an additional term for the reward. We first collect beliefs from both ends of the graphical model sending to time τ, and finding the a_τ there that maximizes (6). If we write out the belief propagation explicitly, it becomes the familiar forward-backward algorithm. For example, the forward message is

$$\begin{aligned} &\mathop{m}_{f_t \to s_{t+1}}(s_{t+1}) \\ &= \sum_{\substack{s_{0:t}, u_{1:t+1}, \\ a_{0:t}, o_{1:t+1}}} \exp \left(\sum_{t'=0}^{t} \boldsymbol{\theta}^\top \boldsymbol{\phi}(t') + \gamma_{t'} \log \frac{r(s_{t'}, a_{t'})}{\gamma_{t'}(T+1)} \right) \\ &= \sum_{s_t, a_t, u_{t+1}} \Bigg[\mathop{m}_{a_t \to f_t}(a_t) \mathop{m}_{s_t \to f_t}(s_t) \mathop{m}_{u_{t+1} \to f_t}(u_{t+1}) \\ &\quad \times \exp \left(\boldsymbol{\theta}_f^\top \boldsymbol{\phi}_f(s_t, s_{t+1}, a_t, u_{t+1}) + \gamma_{t'} \log \frac{r(s_{t'}, a_{t'})}{\gamma_{t'}(T+1)} \right) \Bigg]. \end{aligned}$$

Note that averaging over future actions using the sum-product algorithm is different from conventional POMDP optimization, which seeks to maximize the reward over future system actions. It is also possible in our approach to use a max-product algorithm on a_t while using sum-product on the other variables, to achieve maximization over future system actions. However, in our system, as in [19], the model itself contains a stochastic policy that provides a predictive distribution over future actions; it remains to be seen whether there is a benefit in maximizing over future actions.

Our approach reduces the space complexity from $O(|S||A|)$ to $O(|S|)$. The time complexity is also reduced by the same factor. This is an exact analog to cost-augmented inference in CRFs with edge-factored costs [10]. To our knowledge, the product form (6) has never been discussed in the context of POMDPs except in [19].

3.2. Learning

The learning problem is similar to planning, except that instead of finding the best action we are interested in finding the best model parameters. In other words, we want to find $\boldsymbol{\theta}$ such that the expected reward,

$$\mathcal{R}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{s_{0:T}, a_{0:T}, \\ u_{1:T}}} \left[\frac{1}{T+1} \sum_{t=0}^{T} r(s_t, a_t) \,\middle|\, o_{1:T} \right], \qquad (8)$$

is maximized given all observations $o_{1:T}$. Again the expectation is taken over all variables not given, namely all states, all system actions, and all user actions.

We use gradient descent to optimize the learning objective. It is well known that for log-linear models, the gradient for this type of objective has a nice form. In general, for any utility function $v(\mathbf{x})$ and probability distribution of the form

$$p(\mathbf{x}) = \frac{1}{Z_\theta} \exp\big(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})\big), \text{ where } Z_\theta = \sum_{\mathbf{x}} \exp\big(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})\big), \quad (9)$$

the derivative of the expected utility is:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}}[v(x)] = \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(x) v(x)] - \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}(x)] \mathbb{E}[v(x)]. \qquad (10)$$

Note that for each parameter θ_i in $\boldsymbol{\theta}$, the derivative is simply the covariance between the corresponding feature ϕ_i and the utility. Thus, the parameters corresponding to features that are positively correlated with utility will be increased, while those whose corresponding features are negatively correlated with utility will be decreased.

Applying this to our model gives:

$$\begin{aligned} \frac{\partial \mathcal{R}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \mathop{\mathbb{E}}_{\substack{s_{0:T}, \\ u_{1:T}}} \left[\left(\sum_{t=0}^{T} \boldsymbol{\phi}(t) \right) \left(\sum_{t=0}^{T} \frac{r(s_t, a_t)}{T+1} \right) \right] \\ &\quad - \mathop{\mathbb{E}}_{\substack{s_{0:T}, a_{0:T}, \\ u_{1:T}}} \left[\sum_{t=0}^{T} \boldsymbol{\phi}(t) \right] \mathop{\mathbb{E}}_{\substack{s_{0:T}, a_{0:T}, \\ u_{1:T}}} \left[\sum_{t=0}^{T} \frac{r(s_t, a_t)}{T+1} \right], \end{aligned}$$

where expectations are computed using $p(s_{0:T}, a_{0:T}, u_{1:T} | o_{1:T})$. In the general case, it may be hard to calculate these quantities, but in our case they can be computed efficiently using belief propagation. This is an exact analog to optimizing empirical Bayes risk in a CRF with edge-factored costs [11].

3.3. Particle Belief Propagation

Because the variable spaces are too large to marginalize over, we tackle the problem using particle belief propagation [18], which we briefly review here.

Consider a message $m_{f_t\to s_{t+1}}(s_{t+1})$ passing from factor node f_t to s_{t+1} by marginalizing over s_t, a_t, and u_{t+1}:

$$\underset{f_t\to s_{t+1}}{m}(s_{t+1}) = \sum_{s_t,a_t,u_{t+1}} \Bigg[\underset{a_t\to f_t}{m}(a_t) \underset{s_t\to f_t}{m}(s_t) \underset{u_{t+1}\to f_t}{m}(u_{t+1}) \times \exp\big(\boldsymbol{\theta}_f^\top \boldsymbol{\phi}_f(s_t,s_{t+1},a_t,u_{t+1})\big)\Bigg].$$

If we rewrite the sum with importance sampling, we get

$$\underset{f_t\to s_{t+1}}{m}(s_{t+1}) = \mathbb{E}_{\pi_t} \Bigg[\underset{a_t\to f_t}{m}(a_t) \underset{s_t\to f_t}{m}(s_t) \underset{u_{t+1}\to f_t}{m}(u_{t+1}) \times \frac{\exp\big(\boldsymbol{\theta}_f^\top \boldsymbol{\phi}_f(s_t,s_{t+1},a_t,u_{t+1})\big)}{\pi_t(a_t)\pi_t(u_t)\pi_t(s_t)}\Bigg],$$

for some sampling distribution $\pi_t(a)$, $\pi_t(u)$, $\pi_t(s)$ over which the expectation is computed. We can then approximate it with a sum

$$\underset{f_t\to s_{t+1}}{m}(s_{t+1}) = \frac{1}{N}\sum_{i=1}^{N} \Bigg[\underset{a_t\to f_t}{m}(a_t^{(i)}) \underset{s_t\to f_t}{m}(s_t^{(i)}) \underset{u_{t+1}\to f_t}{m}(u_{t+1}^{(i)}) \times \frac{\exp\big(\boldsymbol{\theta}_f^\top \boldsymbol{\phi}_f(s_t^{(i)},s_{t+1},a_t^{(i)},u_{t+1}^{(i)})\big)}{\pi_t(a_t^{(i)})\pi_t(u_t^{(i)})\pi_t(s_t^{(i)})}\Bigg],$$

over samples $\big\{(s_t^{(1)},a_t^{(1)},u_{t+1}^{(1)}),\ldots,(s_t^{(N)},a_t^{(N)},u_{t+1}^{(N)})\big\}$. The choice of sampling distribution is discussed in Section 4.

4. EXPERIMENTS

The dataset consists of 694 successful dialog sessions, which are collected from users interacting with an existing rule-based system. Dialog sessions are selected examples in which the existing dialog system performed well. The average number of turns, T, for the dataset is 3.4. In order to better understand how the model works, we conduct the experiments in a restricted setting by using ground-truth texts instead of one-best recognition results as our observations to the model. The language for the system is in Japanese, so we use MeCab [20] to segment Japanese characters into terms as a preprocessing step.

The features we use are of the form $\mathbb{1}_{k\in s_t,k'\in s_{t+1},k''\in a_t}$, $\mathbb{1}_{k\in s_t,k'\in s_{t+1},k''\in u_{t+1}}$, $\mathbb{1}_{k\in a_t,k'\in s_{t+1},k''\in u_{t+1}}$, $\mathbb{1}_{k\in u_t,w_{i-1}w_i\in o_t}$ as discussed in Section 2.2, except that we do not look at production rules that are more than two steps away from the root of the tree in order to avoid overfitting.

The grammars for states are designed according to the functionality of the existing rule-based system. The grammar for system actions additionally contains production rules for requesting actions from the users, and the grammar for user actions contains other additional rules for filling slots and confirming. There are a total of 171 production rules for states, which can generate a total of 1484 distinct parse trees without expanding all the slot variables.

The reward function we use in a given dialog session only depends on the current action a_t, and not on the current state s_t. We define the goal of a successful dialog from the dataset as the last action performed in the dialog session. If the current action matches the goal, then the reward is 1. If the current action matches any actions *en route* to the goal, then the reward is 0.5. Otherwise, the reward is 0.01. In this context, it is important to marginalize over the system actions in the learning objective, so that the system can propose its own actions. Otherwise, if all the actions were observed, the objective function would evaluate to a constant.

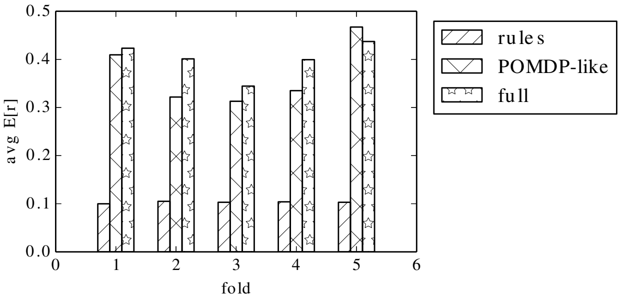

Fig. 4. Five-fold cross-validation results comparing expected reward of our model (with full feature set) to a simple rule-based model (baseline) and a POMDP-like version of our model in which a subset of the features are not used. Our proposed system outperforms the rule-based baseline system by a wide margin and outperforms the POMDP-like model on all but one fold.

For comparison, we implement a simple hand-crafted rule-based system in our framework by setting nonzero parameters only for features corresponding to the state transitions of the existing rule-based system. In addition, we construct a POMDP-like model for comparison, by choosing the largest possible subset of our features so that they act as $p(s_{t+1}|s_t,a_t)$, $p(a_t|s_t)$, and $p(u_{t+1}|a_t,s_{t+1})$ to mimic POMDP.

For evaluation, we train using stochastic gradient descent on the learning objective and compute the expected reward on the test set using five-fold cross-validation. Our sampling distribution for particle belief propagation is uniform for each production rule in the grammar. The marginalization of O for the factor g_t, for simplicity, was assumed to be uniform for all t and all $u \in U$. The number of samples used for each variable was 100. The step size for gradient descent was 100. The number of samples and step size were tuned on a small subset of our dataset. The comparison of the two for the last epoch is shown in Fig. 4.

Across the five folds, both log-linear models achieve higher expected rewards than the rule-based system. In addition, using the full feature set almost always outperforms its POMDP-like counterpart.

The results are promising in that the system performance approaches the maximum possible expected reward for the test set. Although these experiments do not test the planning algorithm, they do test the model predictions, demonstrating that they score almost as well on the test set as the ground-truth actions.

5. CONCLUSION

We present a dialog manager based on a log-linear probabilistic model. We use context-free grammars to impart hierarchical structure to our variables and features. A variational bound on the reward function allows us to perform inference with a single pass of a sum-product algorithm. To handle the large hypothesis space, we use a particle belief propagation method that exploits the grammar's structure. Future work will investigate use of the grammar for efficient re-sampling, as well as maximization over future system actions.

6. REFERENCES

[1] Steve Young, M Gašić, Blaise Thomson, and Jason D Williams, "POMDP-based statistical spoken dialog systems: A review," 2013.

[2] Nicholas Roy, Joelle Pineau, and Sebastian Thrun, "Spoken dialogue management using probabilistic reasoning," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2000, pp. 93–100.

[3] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu, "The hidden information state model: a practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, 2010.

[4] Trung H Bui, BW van Schooten, and DHW Hofs, "Practical dialogue manager development using pomdps," The Association for Computational Linguistics, 2007.

[5] Bo Zhang, Qingsheng Cai, Jianfeng Mao, Eric Chang, and Baining Guo, "Spoken dialogue management as planning and acting under uncertainty.," in *INTERSPEECH*, 2001, pp. 2169–2172.

[6] Cheongjae Lee, Sangkeun Jung, Kyungduk Kim, Donghyeon Lee, and Gary Geunbae Lee, "Recent approaches to dialog management for spoken dialog systems.," *JCSE*, vol. 4, no. 1, pp. 1–22, 2010.

[7] Jason D Williams, Pascal Poupart, and Steve Young, "Factored partially observable markov decision processes for dialogue management," in *4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 76–82.

[8] Milica Gašić, Filip Jurčíček, Blaise Thomson, and Steve Young, "Optimisation for POMDP-based spoken dialogue systems," in *Data-Driven Methods for Adaptive Spoken Dialogue Systems*, pp. 75–101. Springer, 2012.

[9] John Lafferty, Andrew McCallum, and Fernando CN Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML '01: Proceedings of the Eighteenth Int. Conf. on Machine Learning*, 2001, pp. 282–289.

[10] Kevin Gimpel and Noah A Smith, "Softmax-margin CRFs: Training log-linear models with cost functions," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 733–736.

[11] Ying Xiong, Jie Zhu, Hao Huang, and Haihua Xu, "Minimum tag error for discriminative training of conditional random fields," *Information Sciences*, vol. 179, no. 1, pp. 169–179, 2009.

[12] Koichiro Yoshino, Shinji Watanabe, Jonathon Le Roux, and John R. Hershey, "Statistical dialogue management using intention dependency graph," in *IJCNLP*, 2013.

[13] Jason D Williams, "The best of both worlds: unifying conventional dialog systems and POMDPs.," in *INTERSPEECH*, 2008, pp. 1173–1176.

[14] Jason D Williams, "Incremental partition recombination for efficient tracking of multiple dialog states," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5382–5385.

[15] Jesse Hoey and Pascal Poupart, "Solving POMDPs with continuous or large discrete observation spaces," in *International Joint Conference on Artificial Intelligence*. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005, vol. 19, p. 1332.

[16] Andrew Y Ng and Michael Jordan, "PEGASUS: A policy search method for large MDPs and POMDPs," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000, pp. 406–415.

[17] Nicholas Roy and Geoffrey J Gordon, "Exponential family PCA for belief compression in POMDPs," in *Advances in Neural Information Processing Systems*, 2002, pp. 1635–1642.

[18] Alexander T Ihler and David A McAllester, "Particle belief propagation," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 256–263.

[19] Marc Toussaint, Amos Storkey, and Stefan Harmeling, "Expectation-maximization methods for solving (PO) MDPs and optimal control problems," 2010.

[20] "Mecab: Yet another part-of-speech and morphological analyzer," `http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html`.