Probabilistic Dialogue Models with Prior Domain Knowledge

Pierre Lison
Department of Informatics
University of Oslo, Norway
plison@ifi.uio.no

Abstract

Probabilistic models such as Bayesian Networks are now in widespread use in spoken dialogue systems, but their scalability to complex interaction domains remains a challenge. One central limitation is that the state space of such models grows exponentially with the problem size, which makes parameter estimation increasingly difficult, especially for domains where only limited training data is available. In this paper, we show how to capture the underlying structure of a dialogue domain in terms of *probabilistic rules* operating on the dialogue state. The probabilistic rules are associated with a small, compact set of parameters that can be directly estimated from data. We argue that the introduction of this abstraction mechanism yields probabilistic models that are easier to learn and generalise better than their unstructured counterparts. We empirically demonstrate the benefits of such an approach learning a dialogue policy for a human-robot interaction domain based on a Wizard-of-Oz data set.

1 Introduction

Spoken dialogue systems increasingly rely on probabilistic models at various stages of their pipeline. Statistical methods have notably been applied to tasks such as disfluency detection (Lease et al., 2006), semantic parsing (Erdogan et al., 2002; He and Young, 2005), dialogue act recognition (Stolcke et al., 2000; Lan et al., 2008), dialogue management (Frampton and Lemon, 2009; Young et al., 2010), natural language generation (Oh and Rudnicky, 2002; Lemon, 2011) and speech synthesis (Zen et al., 2009).

There are two compelling reasons for this growing interest in statistical approaches: first, spoken dialogue is pervaded with noise and uncertainty (due to e.g. speech recognition errors, linguistic and pragmatic ambiguities, and unknown user intentions), which must be dealt with at all processing stages. Second, a decisive advantage of probabilistic models lies in their ability to be automatically optimised from data, enabling statistically-based dialogue systems to exhibit conversational behaviours that are often more robust, flexible and adaptive than hand-crafted systems (Lemon and Pietquin, 2007).

Despite their success, the use of probabilistic models also presents a number of challenges. The most pressing issue is the paucity of appropriate data sets. Stochastic models often require large amounts of training data to estimate their parameters – either directly (Henderson et al., 2008) or indirectly by way of a user simulator (Schatzmann et al., 2007; Cuayáhuitl et al., 2010). Unfortunately, real interaction data is scarce, expensive to acquire, and difficult to transfer from one domain to another. Moreover, many dialogue domains are inherently open-ended, which means they are not limited to the completion of a single task with predefined features but have to represent a varying number of tasks, complex user models and a rich, dynamic environment. Examples of such domains include human-robot interaction (Kruijff et al., 2010), cognitive assistants and companions (Nguyen, 2005; Cavazza et al., 2010), and tutoring systems (Litman and Silliman, 2004; Eskenazi, 2009). In such settings, the dialogue system might need to track a large number of variables in the course of the interaction, which quickly leads to a combinatorial explosion of the state space.

There is an extensive body of work in the machine learning and planning literature that shows how to address this issue by relying on more expressive representations, able to capture relevant aspects of the problem *structure* in a compact manner. By taking advantage of hierarchical or relational abstractions, system developers can leverage their domain knowledge to yield probabilistic models that are easier to learn (due to a reduced number of parameters) and more efficient to use (since the structure can be exploited by the inference algorithm).

The contributions of this paper are twofold. We first present a new framework for encoding prior knowledge in probabilistic dialogue models, based on the concept of *probabilistic rules*. The framework is very general and can accommodate a wide spectrum of domains and learning tasks, from fully statistical models with virtually no prior knowledge to manually designed models with only a handful of parameters. Second, we demonstrate how this framework can be exploited to learn stochastic dialogue policies with limited data sets using a Bayesian learning approach.

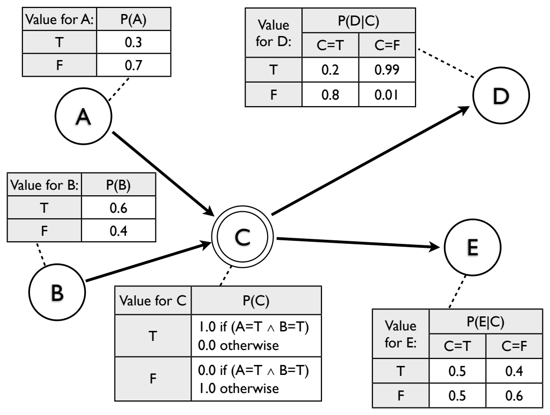
The following pages spell out the approach in more detail. In Section 2, we provide the general background on probabilistic models and their use in spoken dialogue systems. We describe in Section 3 how to encode such models via probabilistic rules and estimate their parameters from data. In Section 4, we detail the empirical evaluation of our approach in a human-robot interaction domain, given small amounts of data collected in Wizard-of-Oz experiments. Finally, we discuss and compare our approach to related work in Section 5.

2 Background

2.1 Bayesian Networks

The probabilistic models used in this paper are expressed as directed graphical models, also known as Bayesian Networks. Let $X_1 ... X_n$ denote a set of random variables. Each variable X_i is associated with a range of mutually exclusive values. In dialogue models, this range is often discrete and can be explicitly enumerated: $Val(X_i) = \{x_i^1, ..., x_i^m\}$.

A Bayesian Network defines the joint probability distribution $P(X_1 ... X_n)$ via conditional dependencies between variables, using a directed graph where each node corresponds to a variable X_i. Each



Value for A:	P(A)
T	0.3
F	0.7

Value for B:	P(B)
T	0.6
F	0.4

Value for C	P(C)
T	1.0 if (A=T ∧ B=T) 0.0 otherwise
F	0.0 if (A=T ∧ B=T) 1.0 otherwise

Value for D:	P(D|C)	
	C=T	C=F
T	0.2	0.99
F	0.8	0.01

Value for E:	P(E|C)	
	C=T	C=F
T	0.5	0.4
F	0.5	0.6

Figure 1: Example of Bayesian network with 5 nodes. The double circles denote a deterministic node. As an example, the query $P(A|D{=}\mathrm{T})$ gives the result $P(A{=}\mathrm{T}|D{=}\mathrm{T}) \approx 0.18$ and $P(A{=}\mathrm{F}|D{=}\mathrm{T}) \approx 0.82$.

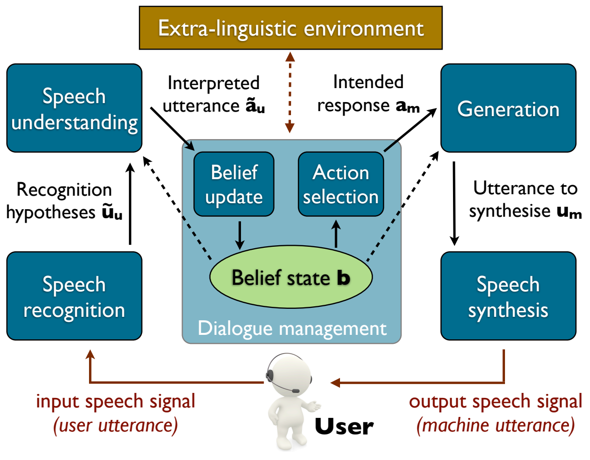
edge $X_i \rightarrow X_j$ denotes a conditional dependence between the two nodes, in which case X_i is said to be a *parent* of X_j. A conditional probability distribution $P(X_i|Par(X_i))$ is associated with each node X_i, where $Par(X_i)$ denotes the parents of X_i.

Conditional probability distributions (CPDs) can be defined in various ways, from look-up tables to deterministic distributions (Koller and Friedman, 2009). Together with the directed graph, the CPDs fully determine the joint probability distribution of the Bayesian Network. The network can be used for inference by querying the distribution of a subset of variables, often given some additional evidence, as illustrated by the example in Figure 1.

2.2 Dialogue Models

A dialogue state $\mathbf{s}$ is usually decomposed into a set of state variables $\mathbf{s} = \{s_1, ... s_n\}$ representing relevant features of the interaction. For instance, the state variables for a human-robot interaction scenario might be composed of tasks to accomplish, the interaction history, past events, as well as objects, spatial locations and agents in the environment.

Given the uncertainty present in spoken dialogue, many variables are only partially observable. We thus encode our knowledge of the current state in a distribution $\mathbf{b}(\mathbf{s}) = P(s_1, ..., s_n)$ called the *belief state*, which can be conveniently expressed as a Bayesian Network (Thomson and Young, 2010). This belief state $\mathbf{b}$ is regularly updated as new infor-



Extra-linguistic environment
Speech understanding
Interpreted utterance $\tilde{\mathbf{a}}_\mathbf{u}$
Intended response $\mathbf{a}_\mathbf{m}$
Generation
Belief update
Action selection
Recognition hypotheses $\tilde{\mathbf{u}}_\mathbf{u}$
Utterance to synthesise $\mathbf{u}_\mathbf{m}$
Belief state $\mathbf{b}$
Speech recognition
Speech synthesis
Dialogue management
input speech signal (user utterance)
User
output speech signal (machine utterance)

Figure 2: Dialogue system architecture schema.

mation becomes available. As illustrated in Figure 2, the whole system pipeline can be formalised in terms of inference steps over this belief state:

1. Upon detection of a new utterance, the speech recogniser generates the N-best list of recognition hypotheses $\tilde{\mathbf{u}}_\mathbf{u} = P(u_u|o)$;
2. Speech understanding then searches for the most likely dialogue act(s) realised in the utterance: $\tilde{\mathbf{a}}_\mathbf{u} = P(a_u|\tilde{\mathbf{u}}_\mathbf{u}, \mathbf{b})$;
3. The belief state is updated with the new interpreted dialogue act: $\mathbf{b}' = P(\mathbf{s}'|\tilde{\mathbf{a}}_\mathbf{u}, \mathbf{b})$;
4. Based on the updated belief state, the action selection searches for the optimal system action to perform: $a^*_m = \arg\max_{a_m} Q(a_m|\mathbf{b})$;
5. The system action is then realised in an utterance u_m, which is again framed as a search for $u^*_m = \arg\max_{u_m} Q(u_m|\mathbf{b}, a_m)$;
6. Finally, the dialogue state is re-updated given the system action: $\mathbf{b}' = P(\mathbf{s}'|a_m, \mathbf{b})$.

The models defined above use $P(x|\mathbf{b})$ as a notational convenience for $\sum_{\mathbf{s^i} \in Val(\mathbf{s})} P(x|\mathbf{s}{=}\mathbf{s^i})\mathbf{b}(\mathbf{s^i})$. The same holds for the estimated values $\tilde{\mathbf{u}}_\mathbf{u}$ and $\tilde{\mathbf{a}}_\mathbf{u}$: $P(x|\tilde{\mathbf{y}}) = \sum_{y^i \in Val(\tilde{\mathbf{y}})} P(x|y{=}y^i)P(y{=}y^i)$.

3 Approach

The starting point of our approach is the observation that dialogue often exhibits a fair amount of *internal structure*. This structure can take several forms.

We can first note that the probability or utility of a given output variable often depends on only a small subset of input variables, although the number and identity of these variables might naturally differ from action to action. The state variable encoding the physical location of a mobile robot is for instance relevant for answering a user requesting its location, but not for responding to a greeting act.

Moreover, the values of the dependent variables can often be grouped into *partitions* yielding similar outcomes, thereby reducing the problem dimensionality. The partitions can generally be expressed via logical conditions on the variable values. As illustration, consider a dialogue where the user can ask yes/no questions pertaining to the colour of specific objects (e.g. “Is the ball red?”). The utility of the system action $\mathrm{Confirm}$ depends on two variables: the user dialogue act, for instance $a_u{=}\ \mathrm{VerifyColour(ball, red)}$, and the object colour, such as $\mathrm{ball.colour} = \mathrm{blue}$. The combination of these two variables can take a wide range of values, but the utility of $\mathrm{Confirm}$ only depends on two partitions: $(\mathrm{VerifyColour(x, y)} \land \mathrm{x.colour}{=}\mathrm{y})$, in which case the utility is positive, and $(\mathrm{VerifyColour(x, y)} \land \mathrm{x.colour}{\neq}\mathrm{y})$, in which case it is negative.

We outline below a generic description framework for expressing this internal structure, based on the concept of *probabilistic rules*. The rules express the distribution of a dialogue model in terms of structured mappings between input and output variables. At runtime, the rules are then combined to perform inference on the dialogue state, i.e. to compute the distribution of the output variables given the input variables. As we shall see, this is done by instantiating the rules and their associated variables to construct an equivalent Bayesian Network used for inference. The probabilistic rules thus function as high-level *templates* for a classical probabilistic model. The major benefit of this approach is that the rule structure is described in exponentially fewer parameters than its plain counterpart, and is thus much easier to learn and to generalise to unseen data.

3.1 Definitions

A probabilistic rule is defined as a condition-effect mapping, where each condition is mapped to a set of alternative effects, each being assigned a distinct probability. The list of conditions is ordered and takes the form of a "**if** ... **then** ... **else**" case expressing the distribution of the output variables depending on the inputs.

Formally, a rule r is defined as an ordered list of cases $\langle c_1, ...c_n\rangle$, where each case c_i is associated with a condition ϕ_i and a distribution over stochastic effects $\{(\psi_i^1, p_i^1), ..., (\psi_i^k, p_i^k)\}$, where ψ_i^j is a stochastic effect and probability $p_i^j = P(\psi_i^j | \phi_i)$, where $p_i^{1...k}$ satisfy the usual probability axioms. The rule reads as such:

$$\begin{array}{l} \textbf{if } (\phi_1) \textbf{ then} \\ \quad \{P(\psi_1^1) = p_1^1, \ ... \ P(\psi_1^k) = p_1^k\} \\ ... \\ \textbf{else if } (\phi_n) \textbf{ then} \\ \quad \{P(\psi_n^1) = p_n^1, \ ... \ P(\psi_n^m) = p_n^m\} \end{array}$$

A final **else** case is implicitly added to the bottom of the list, and holds if no other condition applies. If not overridden, the default effect associated to this last case is void – i.e. it causes no changes to the distribution over the output variables.

Conditions

The rule conditions are expressed as logical formulae grounded in the input variables. They can be arbitrarily complex formulae connected by conjunction, disjunction and negation. The conditions on the input variables can be seen as providing a compact partitioning of the state space to mitigate the dimensionality curse. Without this partitioning in alternative conditions, a rule ranging over m variables each of size n would need to enumerate n^m possible assignments. The partitioning with conditions reduces this number to p mutually exclusive partitions, where p is usually small.

Effects

The rule effects are defined similarly: given a condition holding on a set of input variables, the associated effects define specific *value assignments* for the output variables. The effects can be limited to a single variable or range over several output variables. For action selection, effects can also take the form of assignments of utility values for a particular action, i.e. $Q(a_m = x) = y$, where y is the scalar value for the utility of action x.

Each effect is assigned a probability, and several alternative stochastic effects can be defined for the same case. If a unique effect is specified, it is then implicitly assumed to hold with probability 1.0. The probabilities of stochastic effects and the action utilities are treated as parameters, which can be either hand-coded or estimated from data.

Example

The rules r_1 and r_2 below express the utilities of two actions: the physical action ExecuteMov(X) (with X representing the movement type), and the clarification request AskRepeat.

$$\begin{array}{ll} r_1: & \textbf{if } (a_u = \text{RequestMov(X)}) \textbf{ then} \\ & \{Q(a_m = \text{ExecuteMov(X)}) = \theta_{r_1}^{(1)}\} \\ & \\ r_2: & \textbf{if } (a_u \neq \emptyset \wedge a_m \neq \text{AskRepeat}) \textbf{ then} \\ & \{Q(a_m = \text{AskRepeat}) = \theta_{r_2}^{(1)}\} \\ & \textbf{else if } (a_u \neq \emptyset) \textbf{ then} \\ & \{Q(a_m = \text{AskRepeat}) = \theta_{r_2}^{(2)}\} \end{array}$$

Rule r_1 specifies that, if the last user action a_u is equal to RequestMov(X) (i.e. requesting the robot to execute a particular movement X), the utility associated with ExecuteMov(X) is equal to the parameter $\theta_{r_1}^1$. Similarly, the rule r_2 specifies the utility of the clarification request AskRepeat, provided that the last user action a_u is assigned to a value (i.e. is different than $\emptyset$). Two cases are distinguished in r_2, depending on whether the previous system action was already $AskRepeat$. This partitioning enables us to assign a distinct utility to the clarification request if one follows the other, in order to e.g. penalise for the repeated clarification.

As illustration, assume that $\theta_{r_1}^{(1)} = 2.0$, $\theta_{r_2}^{(1)} = 1.3$, $\theta_{r_2}^{(2)} = 1.1$, and that the belief state contains a state variable a_u with the following distribution:

$$\begin{array}{ll} P(a_u = \text{RequestMov(LiftBothArms)}) & = 0.7 \\ P(a_u = \text{RequestMov(LiftLeftArm)}) & = 0.2 \\ P(a_u = \emptyset) & = 0.1 \end{array}$$

The optimal system action in this case is therefore $\text{ExecuteMov(LiftBothArms)}$ with utility 1.4, followed by AskRepeat with utility 1.17, and $\text{ExecuteMov(LiftLeftArm)}$ with utility 0.4.

3.2 Inference

Given a belief state $\mathbf{b}$, we perform inference by constructing a Bayesian Network corresponding to the application of the rules. Algorithm 1 describes the construction procedure, which operates as follows:

1. We initialise the Bayesian Network with the variables in the belief state;
2. For every rule r in the rule set, we create a condition node ϕ_r and include the conditional dependencies with its input variables;
3. We create an effect node ψ_r conditioned on ϕ_r, expressing the possible effects of the rule;
4. Finally, we create the (chance or value) nodes corresponding to the output variables of the rule, as specified in the effects.

Rule r_2 described in the previous section would for instance be translated into a condition node ϕ_{r_2} with 3 values (corresponding to the specified conditions and a default **else** condition if none applies) and an effect node ψ_{r_2} also containing 3 values (the two specified effects and a void effect associated with the default condition). Figure 3 illustrates the application of rules r_1 and r_2.

Once the Bayesian network is constructed, queries can be evaluated using any standard algorithm for exact or approximate inference. The procedure is an instance of *ground inference* (Getoor and Taskar, 2007), since the rule structure is grounded in a standard Bayesian Network.

3.3 Parameter Learning

The estimation of the rule parameters can be performed using a Bayesian approach by adding parameter nodes $\boldsymbol{\theta} = \theta_1 ... \theta_k$ to the Bayesian Network,

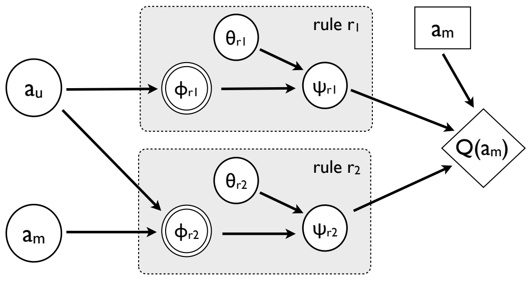


Figure 3: Bayesian Network with the rules r_1 and r_2.

and updating their distribution given a collection of training data. Each data sample d is a pair $(\mathbf{b}_d, t_d)$, where $\mathbf{b}_d$ is the belief state for the specific sample, and t_d the target value. The target value depends on the model to learn – for learning dialogue policies, it corresponds to the selected action a_m.

Algorithm 1 : NETWORKCONSTRUCTION $(\mathbf{b}, \mathcal{R})$

```
Require: b: Current belief state
Require: R: Set of probabilistic rules
 1: $\mathcal{B} \leftarrow \mathbf{b}$
 2: for all rule $r \in \mathcal{R}$ do
 3:     $\mathcal{I}_r \leftarrow$ INPUTNODES$(r)$
 4:     $\phi_r \leftarrow$ CONDITIONNODE$(r)$
 5:     Add $\phi_r$ and dependencies $\mathcal{I}_r \rightarrow \phi_r$ to $\mathcal{B}$
 6:     $\psi_r \leftarrow$ EFFECTNODE$(r)$
 7:     Add $\psi_r$ and dependency $\phi_r \rightarrow \psi_r$ to $\mathcal{B}$
 8:     $\mathcal{O}_r \leftarrow$ OUTPUTNODES$(r)$
 9:     for all output variable $o \in \mathcal{O}_r$ do
10:         Add/modify node $o$ and dep. $\psi_r \rightarrow o$ to $\mathcal{B}$
11:     end for
12: end for
13: return $\mathcal{B}$
```

Algorithm 2 : PARAMETERLEARNING $(\mathcal{R}, \boldsymbol{\theta}, \mathcal{D})$

```
Require: $\mathcal{R}$: Set of probabilistic rules
Require: $\boldsymbol{\theta}$: Parameters with prior distribution
Require: $\mathcal{D}$: Training sample
 1: for all data $d \in \mathcal{D}$ do
 2:     $\mathcal{B} \leftarrow$ NETWORKCONSTRUCTION$(\mathbf{b}_d, \mathcal{R})$
 3:     Add parameters nodes $\boldsymbol{\theta}$ to $\mathcal{B}$
 4:     for all $\theta_i \in \boldsymbol{\theta}$ do
 5:         $P(\theta'_i|d) = \alpha\, P(t_d|\mathbf{b}_d, \theta_i)\, P(\theta_i)$
 6:     end for
 7: end for
 8: return $\boldsymbol{\theta}$
```

To estimate the parameters $\boldsymbol{\theta}$, we start from an initial prior distribution. Then, for each sample d in the training data, we construct the corresponding Bayesian Network from its belief state $\mathbf{b}_d$ and the rules, including nodes corresponding to the unknown rule parameters. Then, for each parameter θ_i, we compute its posterior distribution given the data (Koller and Friedman, 2009):

$$P(\theta'_i|d) = \alpha\, P(t_d|\mathbf{b}_d, \theta_i)\, P(\theta_i) \qquad (1)$$

Given the number of parameters in our example domain and their continuous range, we used approximate inference to calculate the posterior efficiently, via direct sampling from a set of parameter values. The constant α serves as a normalisation factor over the sampled parameter values for θ_i. The procedure is repeated for every sample, as shown in Algorithm 2. The parameter distribution will thus progressively narrow down its spread to the values providing the best fit for the training data.

4 Evaluation

We evaluated our approach in the context of a dialogue policy learning task for a human-robot interaction scenario. The main question we decided to address is the following: how much does the rule structure contribute to the parameter estimation of a given probabilistic model, especially for domains with limited amounts of available data? The objective of the experiment was to learn the rule parameters corresponding to the policy model $Q(a_m|\mathbf{s})$ from a Wizard-of-Oz data collection. In this particular case, the parameters correspond to the utilities of the various actions. The policy model used in the experiment included a total of 14 rules.

We compared our approach with two baselines which are essentially "flattened" or rolled-out versions of the rule-based model. The input and output variables remain identical, but they are directly connected, without the ϕ and ψ nodes serving as intermediate structures. The two baselines are (1) a plain multinomial model and (2) a linear model of the input variables. We are thus comparing three versions of the $Q(a_m|\mathbf{s})$ model: two baselines where a_m is directly dependent on the state variables, and our approach where the dependency is realised indirectly through condition and effect nodes.

4.1 Experimental Setup

The scenario for the Wizard-of-Oz experiment involved a human user and a Nao robot[1] (see Figure 4). The user was instructed to teach the robot a sequence of basic movements (lift the left arm, step forward, kneel down, etc.) using spoken commands. The interaction included various dialogue acts such as clarification requests, feedbacks, acknowledgements, corrections, etc. Short examples of recorded dialogues are provided in the appendix.

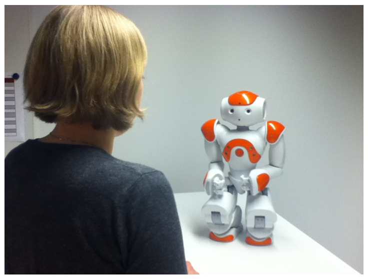


Figure 4: Human user interacting with the Nao robot.

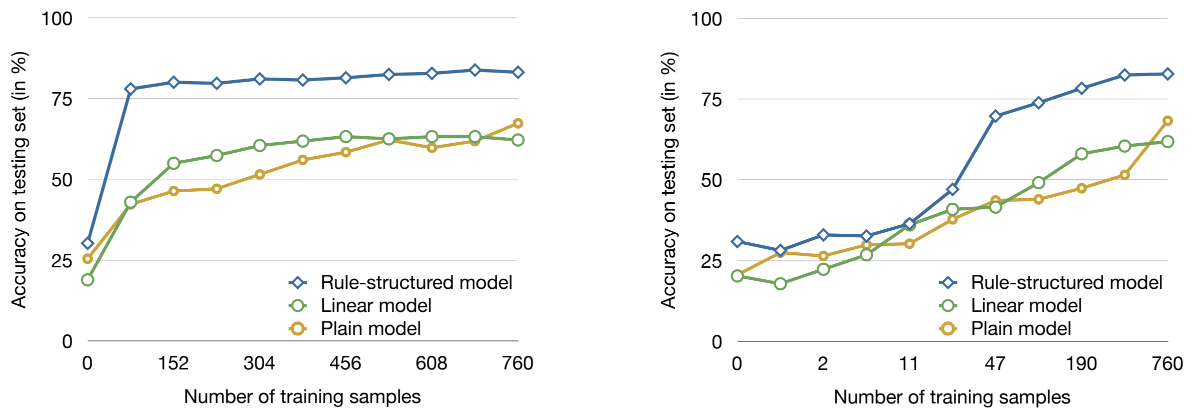
In addition to the policy model, the dialogue system include a speech recognizer (Vocon 3200 from Nuance) connected to the robot microphones, shallow components for dialogue act recognition and generation, a text-to-speech module, and components for planning the robot movements and controlling its motors in real-time. All components are connected to the shared belief state, and read/write to it as they process their data flow.

We collected a total of 20 interactions with 7 users and one wizard playing the role of the policy model, for a total of 1020 system turns, summing to around 1h of interaction. All the interactions were performed in English. The wizard only had access to the N-best list output from the speech recogniser, and could then select which action to perform from a list of 14 alternatives (such as $\mathrm{AskRepeat}$, $\mathrm{DemonstrateMove}$, $\mathrm{UndoMove}$, $\mathrm{AskForConfirmation}$, etc). Each selected action was recorded along with the belief state (including the full probability distribution for every state variable) in effect at the time of the selection.

4.2 Analysis

The data set was split into training (75% of the system turns) and test data (remaining 25%) used to measure the accuracy of our policies. The accuracy is defined as the percentage of actions corresponding to the gold standard action selected by the wizard. The parameter distributions are initialised with uniform priors, and are progressively refined as more data points are processed. We calculated the accuracy by sampling over the parameters, performing inference over the resulting models, and finally averaging over the inference results.

[1]A programmable humanoid robot developed by Aldebaran Robotics, `http://www.aldebaran-robotics.com`.



(a) Linear scale (b) Log-2 scale

Figure 5: Learning curves for the overall accuracy of the learned dialogue policy, on a held-out test set of 255 actions, depending on the size of the training sample. The accuracy results are given for the plain, linear and rule-structured policy models, using linear (left) and logarithmic scales (right).

Type of model	*Accuracy (in %)*
Plain model	67.35
Linear model	61.85
Rule-structured model	**82.82**

Table 1: Accuracy results for the three action selection models on a test set, using the full training set.

Table 1 provides the accuracy results. The differences between our model and the baselines are statistically significant using Bonferroni-corrected paired t-tests, with p-value < 0.0001. The 17% of actions labelled as incorrect are mainly due to the high degree of noise in the data set, and the sometimes inconsistent or unpredictable behaviour of the wizard (regarding e.g. clarification requests).

It is instructive to analyse the learning curve of the three models, shown in Figure 5. Given its smaller number of parameters, the rule-structured model is able to converge to near-optimal values after observing only a small fraction of the training set. As the figure shows, the baseline models do also improve their accuracies over time, but at a much slower rate. The linear model is comparatively faster than the plain model, but levels off towards the end, possibly due to the non-linearity of some dialogue strategies. The plain model continues its convergence and would probably reach an accuracy similar to the rule-structured model if given much larger amounts of training data. Note that since the parameters are initially uniformly distributed, the accuracy is already non-zero before learning, since a random assignment of parameters has a low but non-zero chance of leading to the right action.

5 Discussion and Related Work

The idea of using structural knowledge in probabilistic models has been explored in many directions, both in the fields of decision-theoretic planning and of reinforcement learning (Hauskrecht et al., 1998; Pineau, 2004; Lang and Toussaint, 2010; Otterlo, 2012) and in statistical relational learning (Jaeger, 2001; Richardson and Domingos, 2006; Getoor and Taskar, 2007). The introduced structure may be hierarchical, relational, or both. As in our approach, most of these frameworks rely on the use of expressive representations as *templates* for grounded probabilistic models.

In the dialogue management literature, most structural approaches rely on a clear-cut task decomposition into goals and sub-goals (Allen et al., 2000; Steedman and Petrick, 2007; Bohus and Rudnicky, 2009), where the completion of each goal is assumed to be fully observable, discarding any remaining uncertainty. Information-state approaches to dialogue management (Larsson and Traum, 2000; Bos et al., 2003) also rely on a shared state updated according to a rich repository of rules, but contrary to the approach presented here, these rules are generally deterministic and do not include learnable parameters.

The literature on dialogue policy optimisation with reinforcement learning also contains several approaches dedicated to dimensionality reduction for large state-action spaces, such as function approximation (Henderson et al., 2008), hierarchical reinforcement learning (Cuayáhuitl et al., 2010) and summary POMDPs (Young et al., 2010). Most of these approaches rely on large but weakly structured state spaces (generally encoded as large lists of features), which are suited for slot-filling dialogue applications but are difficult to transfer to more open-ended or relational domains. The idea of state space partitioning, implemented here via high-level conditions, has also been explored in recent papers (Williams, 2010; Crook and Lemon, 2010). Finally, Cuayáhuitl (2011) describes a closely-related approach using logic-based representations of the state-action space for relational MDPs. His approach is however based on reinforcement learning with a user simulator, while the learning procedure presented here relies on supervised learning from a limited data set. He also reduced his belief state to fully observable variables, whereas we retain the partial observability associated with each variable.

An important side benefit of structured representations in probabilistic models is their improved readability for human designers, who are able to use these powerful abstractions to encode their prior knowledge of the dialogue domain in the form of pragmatic rules, generic background knowledge, or task-specific constraints. There has been previous work on integrating expert knowledge into dialogue policy learning, using finite-state policies or ad-hoc constraints to filter a plain statistical model (Williams, 2008; Henderson et al., 2008). The approach presented in this paper is however more general since it does not rely on an external filtering mechanism but directly incorporates prior domain knowledge into the statistical model.

6 Conclusions

We showed in this paper how to represent the underlying structure of probabilistic models for dialogue using *probabilistic rules*. These rules are defined as structured mappings over variables of the dialogue state, specified using high-level conditions and effects. These rules can include parameters such as effect probabilities or action utilities. Probabilistic rules allow the system designer to exploit powerful generalisations in the dialogue domain specification without sacrificing the probabilistic nature of the model. The framework is very general and can express a wide spectrum of models, from classical models fully estimated from data to ones incorporating rich prior knowledge. The choice of model within this spectrum is therefore essentially a design decision dependent on the relative availabilities of training data and domain knowledge.

We have also presented algorithms for constructing Bayesian Networks corresponding to the application of the rules and for estimating their parameters from data using Bayesian inference. The presented approach has been implemented in a spoken dialogue system for human-robot interaction, and validated on a policy learning task based on a Wizard-of-Oz data set. The empirical results have shown that the rule structure enables the learning algorithm to converge faster and with better generalisation performance.

We are currently working on extending this approach in two directions. First, we would like to extend our parameter estimation method to Bayesian model-based reinforcement learning. The current implementation operates in a supervised learning mode, which requires expert data. Alternatively, one could estimate the model parameters in a fully online fashion, without any supervisory input, by incorporating model uncertainty into the inference and continuously adapting the parameter distribution from (real or simulated) interaction experience, using the same Bayesian approach we have outlined in this paper (Ross et al., 2011).

The second direction is the extension of our work to tasks other than action selection. The framework we have presented is not confined to dialogue policy learning but can be used to structure any probabilistic model[2]. It is therefore possible to use probabilistic rules as a unifying framework for all models defined in a given architecture, and exploit it to perform *joint optimisation* of dialogue understanding, action selection and generation.

[2]In fact, the dialogue understanding and generation models used for the evaluation were already structured with probabilistic rules, but with fixed, hand-crafted parameters.

Acknowledgements

The author would like to thank Stephan Oepen, Erik Velldal and Amanda Stent for useful comments on an earlier version of this paper.

References

J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. 2000. An architecture for a generic dialogue shell. *Natural Language Engineering*, 6:213–228.

D. Bohus and A. I. Rudnicky. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23:332–361.

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124.

M. Cavazza, R. Santos de la Camara, M. Turunen, J. Relaño-Gil, J. Hakulinen, N. Crook, and D. Field. 2010. How was your day? an affective companion ECA prototype. In *Proceedings of the 11th SIGDIAL Meeting on Discourse and Dialogue*, pages 277–280.

P. A. Crook and O. Lemon. 2010. Representing uncertainty about complex user goals in statistical dialogue systems. In *Proceedings of the 11th SIGDIAL meeting on Discourse and Dialogue*, pages 209–212.

H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. 2010. Evaluation of a hierarchical reinforcement learning spoken dialogue system. *Computer Speech & Language*, 24:395–429.

H. Cuayáhuitl. 2011. Learning Dialogue Agents with Bayesian Relational State Representations. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI-KRPDS)*, Barcelona, Spain.

H. Erdogan, R. Sarikaya, Y. Gao, and M. Picheny. 2002. Semantic structured language models. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*, Denver, USA.

M. Eskenazi. 2009. An overview of spoken language technology for education. *Speech Commununications*, 51:832–844.

M. Frampton and O. Lemon. 2009. Recent research advances in reinforcement learning in spoken dialogue systems. *Knowledge Engineering Review*, 24(4):375–408.

L. Getoor and B. Taskar. 2007. *Introduction to Statistical Relational Learning*. The MIT Press.

M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier. 1998. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–229.

Y. He and S. Young. 2005. Semantic processing using the hidden vector state model. *Computer Speech & Language*, 19(1):85–106.

J. Henderson, O. Lemon, and K. Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34:487–511.

M. Jaeger. 2001. Complex probabilistic modeling with recursive relational bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):179–220.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

G.-J. M. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, Hendrik Zender, and Ivana Kruijff-Korbayová, 2010. *Situated Dialogue Processing for Human-Robot Interaction*, chapter 8. Springer Verlag, Heidelberg, Germany.

K. C. Lan, K. S. Ho, R. W. Pong Luk, and H. Va Leong. 2008. Dialogue act recognition using maximum entropy. *Journal of the American Society for Information Science and Technology (JASIST)*, pages 859–874.

T. Lang and M. Toussaint. 2010. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39:1–49.

S. Larsson and D. R. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natuarl Language Engineering*, 6(3-4):323–340, September.

M. Lease, M. Johnson, and E. Charniak. 2006. Recognizing disfluencies in conversational speech. *IEEE Transactions on Audio, Speech & Language Processing*, 14(5):1566–1573.

O. Lemon and O. Pietquin. 2007. Machine Learning for Spoken Dialogue Systems. In *Proceedings of the 10th European Conference on Speech Communication and Technologies (Interspeech'07)*, pages 2685–2688.

O. Lemon. 2011. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. *Computer Speech & Language*, 25:210–221.

D. J. Litman and S. Silliman. 2004. ITSPOKE: an intelligent tutoring spoken dialogue system. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL 2004)*, pages 5–8.

A. Nguyen. 2005. An agent-based approach to dialogue management in personal assistants. In *Proceedings of the 2005 International conference on Intelligent User Interfaces (IUI)*, pages 137–144. ACM Press.

A. Oh and A. I. Rudnicky. 2002. Stochastic natural language generation for spoken dialog systems. *Computer Speech & Language*, 16(3-4):387–407.

M. Otterlo. 2012. Solving relational and first-order logical markov decision processes: A survey. In *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 253–292. Springer Berlin Heidelberg.

J. Pineau. 2004. *Tractable Planning Under Uncertainty: Exploiting Structure*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, USA.

M. Richardson and P. Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

S. Ross, J. Pineau, B. Chaib-draa, and P. Kreitmann. 2011. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, 12:1729–1770.

J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL 2007, pages 149–152.

M. Steedman and R. P. A. Petrick. 2007. Planning dialog actions. In *Proceedings of the 8th SIGDIAL Meeting on Discourse and Dialogue*, pages 265–272, Antwerp, Belgium.

A. Stolcke, N. Coccaro, R. Bates, P. Taylor, C. Van Ess-Dykema, K. Ries, E. Shriberg, D. Jurafsky, R. Martin, and M. Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computional Linguistics*, 26:339–373.

V. Thomson and S. Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24:562–588, October.

J. D. Williams. 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *International Conference on Speech and Language Processing (ICSLP 2008)*, Brisbane, Australia.

J. D. Williams. 2010. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5382–5385.

S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24:150–174.

H. Zen, K. Tokuda, and A. W. Black. 2009. Review: Statistical parametric speech synthesis. *Speech Communications*, 51:1039–1064.

A Example of interaction

We present below two recorded excerpts of Wizard-of-Oz interactions. The user utterances are simplified to show only the hypothesis with highest score (indicated in parenthesis) in the N-best list.

Transcript A.1 Wizard-of-Oz interaction 1

HUMAN: Hi Lenny (*0.74*)
ROBOT: Hello there!
HUMAN: Move your left arm up (*0.41*)
ROBOT: Sure!
[*executing movement, waits a few seconds*]
ROBOT: Was that correct?
HUMAN: Yes (*0.88*)
[*registers movement as confirmed*]
HUMAN: Move your right arm up at the same time (*0.38*)
ROBOT: Let's see
[*executing movement*]
ROBOT: Was that correct?
HUMAN: No (*0.50*)
[*goes back to previous position*]
ROBOT: What should I do then?
[...]

Transcript A.2 Wizard-of-Oz interaction 2

HUMAN: Stand up (*0.41*)
ROBOT: Let's see
[*executing movement*]
HUMAN: Move your right arm forward (*0.35*)
[*last move is implicitly grounded*]
ROBOT: Let's see
[*executing movement*]
HUMAN: Lower the left arm (*0.34*)
ROBOT: Sorry, could you repeat?
HUMAN: Lift your left arm (*0.37*)
ROBOT: Sure
[*executing movement, waits a few seconds*]
ROBOT: Should I do it like this?
[...]