Probabilistic Dialogue Models with Prior Domain Knowledge

Pierre Lison
Language Technology Group
University of Oslo

July 6, 2012
SIGDIAL
Introduction

• The use of statistical models is getting increasingly popular in spoken dialogue systems

• But scalability remains a challenge for many domains
Introduction

- The use of *statistical models* is getting increasingly popular in spoken dialogue systems
- But scalability remains a challenge for many domains

<table>
<thead>
<tr>
<th><strong>Advantages</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit account of <em>uncertainties</em>, increased <em>robustness</em> to errors</td>
</tr>
<tr>
<td>Better domain- and user-<em>adaptivity</em>, more <em>natural</em> and <em>flexible</em> conversational behaviours</td>
</tr>
</tbody>
</table>
Introduction

• The use of *statistical models* is getting increasingly popular in spoken dialogue systems

• But scalability remains a challenge for many domains

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit account of <em>uncertainties</em>, increased <em>robustness</em> to errors</td>
<td>Good domain data is <em>scarce</em> and expensive to acquire!</td>
</tr>
<tr>
<td>Better domain- and user-<em>adaptivity</em>, more <em>natural</em> and <em>flexible</em> conversational behaviours</td>
<td><em>Scalability</em> to complex domains (state space grows exponentially with the problem size)</td>
</tr>
</tbody>
</table>
• Well-known problem in A.I. and machine learning

• Solutions typically involve the use of more expressive representations

  • Capturing relevant aspects of the problem structure

  • Taking advantage of hierarchical or relational abstractions

• We present here such an abstraction mechanism, based on the concept of probabilistic rule

• Goal: leverage our prior domain knowledge to yield structured, compact probabilistic models
Key idea

• Observation: dialogue models exhibit a fair amount of *internal structure*:
  
  • Probability (or utility) distributions can often be *factored*
  
  • Even if the full distribution has many dependencies, the probability (or utility) of a *specific outcome* often depends on only a small subset of variables
  
  • Finally, the values of the dependent variables can often be grouped into *partitions*
Example of partitioning

- Consider a dialogue where the user asks a robot yes/no questions about his location.
- The state contains the following variables:
  - Last user dialogue act, e.g. $a_u = \text{AreYouIn(corridor)}$
  - The robot location, e.g. location = kitchen
- You want to learn the utility of $a_m = \text{SayYes}$
Example of partitioning

- Consider a dialogue where the user asks a robot yes/no questions about his location.
- The state contains the following variables:
  - Last user dialogue act, e.g. $a_u = \text{AreYouIn}(\text{corridor})$
  - The robot location, e.g. $\text{location} = \text{kitchen}$
- You want to learn the utility of $a_m = \text{SayYes}$
- The combination of the two variables can take many values, but they can be partitioned in two sets:

  $a_u = \text{AreYouIn}(x) \land \text{location} = x \quad \rightarrow \quad \text{positive utility}$
  $a_u = \text{AreYouIn}(x) \land \text{location} \neq x \quad \rightarrow \quad \text{negative utility}$
Probabilistic rules

• Probabilistic rules attempt to capture such kind of structure

• They take the form of structured if...then...else cases, mappings from conditions to (probabilistic) effects:

\[
\text{if } (\text{condition}_1 \text{ holds}) \text{ then } \\
P(\text{effect}_1) = \theta_1, \quad P(\text{effect}_2) = \theta_2 \\
\text{else if } (\text{condition}_2 \text{ holds}) \text{ then } \\
P(\text{effect}_3) = \theta_3 \\
\]

• For action-selection rules, the effect associates utilities to particular actions:

\[
\text{if } (\text{condition}_1 \text{ holds}) \text{ then } \\
Q(\text{actions}) = \theta_1 \\
\]
• **Conditions** are arbitrary logical formulae on state variables

• **Effects** are value assignments on state variables

• Example of rule for action selection:

\[
\begin{align*}
\text{if } (a_u = \text{AreYouIn}(x) \land \text{location} = x) \text{ then} \\
&\{Q(a_m = \text{SayYes}) = 3.0\} \\
\text{else if } (a_u = \text{AreYouIn}(x) \land \text{location} \neq x) \text{ then} \\
&\{Q(a_m = \text{SayNo}) = 3.0\}
\end{align*}
\]

• Effect probabilities and utilities are *parameters* which can be estimated from data
Rule-based state update

- How are these rules applied in practice?
- The architecture revolves around a shared dialogue state, represented as a **Bayesian network**
- At runtime, the rules are *instantiated* in the network, updating and expanding it with new nodes and dependencies
- The rules thus function as *high-level templates* for a classical probabilistic model
Example
Example

location

office $[P=0.95]$  
kitchen $[P=0.05]$  

$a_u$

AreYouIn(kitchen) $[P=0.7]$  
AreYouIn(corridor) $[P=0.2]$  
None $[P=0.1]$
Example

Rule 1: if \((a_u = \text{AreYouIn}(x) \land \text{location} = x)\) then
\[Q(a_m = \text{SayYes}) = 3.0\]
else if \((a_u = \text{AreYouIn}(x) \land \text{location} \neq x)\) then
\[Q(a_m = \text{SayNo}) = 3.0\]
**Example**

**Rule 1:** if \( a_u = \text{AreYouIn}(x) \land \text{location} = x \) then
\[
\{Q(a_m = \text{SayYes}) = 3.0\}
\]
else if \( a_u = \text{AreYouIn}(x) \land \text{location} \neq x \) then
\[
\{Q(a_m = \text{SayNo}) = 3.0\}
\]
Example

Rule 1: \[ \text{if } (a_u = \text{AreYouIn}(x) \land \text{location} = x) \text{ then} \]
\[ \{ Q(a_m = \text{SayYes}) = 3.0 \} \]
else if \((a_u = \text{AreYouIn}(x) \land \text{location} \neq x) \text{ then} \]
\[ \{ Q(a_m = \text{SayNo}) = 3.0 \} \]

\[ Q(a_m=\text{SayYes}) = 0.105 \]
\[ Q(a_m=\text{SayNo}) = 2.6 \]
Example

location

office \( [P=0.95] \)
kitchen \( [P=0.05] \)

\( \phi_1 \)

\( \psi_1 \)

\( a_u \)

\( a_m \)

AreYouIn(kitchen) \( [P=0.7] \)
AreYouIn(corridor) \( [P=0.2] \)
None \( [P=0.1] \)
Example

Rule 2: if $(a_u \neq \text{None})$ then 
\[
\{ Q(a_m = \text{AskRepeat}) = 0.5 \} 
\]
Rule 2: if \( a_u \neq \text{None} \) then 
\[ Q(a_m = \text{AskRepeat}) = 0.5 \]
Example

Rule 2: if \( a_u \neq \text{None} \) then
\[
\{ Q(a_m = \text{AskRepeat}) = 0.5 \}
\]

location

office \([P=0.95]\)  
kitchen \([P=0.05]\)

\( a_u \)

\( \phi_1 \)

\( \phi_2 \)

\( a_m \)

\( Q(a_m = \text{SayYes}) = 0.105 \)
\( Q(a_m = \text{SayNo}) = 2.6 \)
\( Q(a_m = \text{AskRepeat}) = 0.45 \)
Processing workflow

• The dialogue state, encoded as a Bayesian Network, is the central, shared information repository

• Each processing task (understanding, management, generation, etc.) read and write to it

• Many of these tasks are expressed in terms of collections of probabilistic rules
Parameter learning

- The rule parameters (probabilities or utilities) must be estimated from empirical data
- We adopted a Bayesian approach, where the parameters are themselves defined as variables
- The parameter distributions will then be modified given the evidence from the training data
Parameter learning

• The rule parameters (probabilities or utilities) must be estimated from empirical data

• We adopted a Bayesian approach, where the parameters are themselves defined as variables

• The parameter distributions will then be modified given the evidence from the training data
Parameter learning

- The rule parameters (probabilities or utilities) must be estimated from empirical data.
- We adopted a Bayesian approach, where the parameters are themselves defined as variables.
- The parameter distributions will then be modified given the evidence from the training data.
Evaluation

- Policy learning task in a human-robot interaction scenario, based on Wizard-of-Oz training data
- Objective: estimate the utilities of possible system actions
- Baselines: «rolled-out» versions of the model
  - «plain» probabilistic models with identical input and output variables, but without the condition and effect nodes as intermediary structures
Evaluation

• Policy learning task in a human-robot interaction scenario, based on Wizard-of-Oz training data

• Objective: estimate the utilities of possible system actions

• Baselines: «rolled-out» versions of the model
  • «plain» probabilistic models with identical input and output variables, but without the condition and effect nodes as intermediary structures
Evaluation

• Policy learning task in a human-robot interaction scenario, based on Wizard-of-Oz training data
• Objective: estimate the utilities of possible system actions
• Baselines: «rolled-out» versions of the model
  • «plain» probabilistic models with identical input and output variables, but without the condition and effect nodes as intermediary structures
Experimental setup

- Interaction scenario: users instructed to teach the robot a sequence of basic movements (e.g. a small dance)
- Dialogue system comprising ASR and TTS modules, shallow components for understanding and generation, and libraries for robot control
- The Wizard had access to the dialogue state and took decisions based on it (among a set of 14 alternatives)
- 20 interactions with 7 users, for a total of 1020 turns

Each sample $d$ in the data set is a pair $(b_d, t_d)$:
- $b_d$ is a recorded dialogue state
- $t_d$ is the «gold standard» system action selected by the Wizard at the state $b_d$
Empirical results

- Data set split into training (75%) and testing (25%)
- Accuracy measure: percentage of actions corresponding to the ones selected by the Wizard
  - But Wizard sometimes inconsistent / unpredictable
- The rule-structured model outperformed the two baselines in accuracy and convergence speed

<table>
<thead>
<tr>
<th>Type of model</th>
<th>Accuracy (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain model</td>
<td>67.35</td>
</tr>
<tr>
<td>Linear model</td>
<td>61.85</td>
</tr>
<tr>
<td>Rule-structured model</td>
<td><strong>82.82</strong></td>
</tr>
</tbody>
</table>
Learning curve (linear scale)

Accuracy on testing set (in %) vs. Number of training samples.

- Rule-structured model
- Linear model
- Plain model
Learning curve (log-2 scale)

Accuracy on testing set (in %)

Number of training samples

Rule-structured model
Linear model
Plain model
Conclusions

- **Probabilistic rules** used to capture the underlying structure of dialogue models
- Allow developers to exploit powerful generalisations and domain knowledge without sacrificing the probabilistic nature of the model
- Framework validated on a policy learning task based on a Wizard-of-Oz dataset
- Future work: extend the approach towards model-based Bayesian reinforcement learning