Int J Speech Technol (2014) 17:309–323
DOI 10.1007/s10772-014-9244-6

Dialogue POMDP components (part I): learning states and observations

Hamid R. Chinaei · Brahim Chaib-draa

Received: 9 July 2013 / Accepted: 15 July 2014

Abstract The partially observable Markov decision process (POMDP) framework has been applied in dialogue systems as a formal framework to represent uncertainty explicitly while being robust to noise. In this context, estimating the dialogue POMDP model components is a significant challenge as they have a direct impact on the optimized dialogue POMDP policy. To achieve such an estimation, we propose methods for learning dialogue POMDP model components using noisy and unannotated dialogues. Specifically, we introduce techniques to learn the set of possible user intentions from dialogues, use them as the dialogue POMDP states, and learn a maximum likelihood POMDP transition model from data. Since it is crucial to reduce the observation state size, we then propose two observation models: the keyword model and the intention model. Using these two models, the number of observations is reduced significantly while the POMDP performance remains high particularly in the intention POMDP. Learning states and observations sustaining a POMDP are both covered in this first part (part I) and experimented from dialogues collected by SmartWheeler (an intelligent wheelchair which aims to help persons with disabilities). Part II covers the reward model learning required by the POMDP.

Keywords Partially observable Markov decision processes (POMDP) · Unsupervised learning · Learning observations and states · Healthcare dialogue management

H. R. Chinaei · B. Chaib-draa (✉)
Computer Science Department, Laval University,
Quebec, QC, Canada
e-mail: chaib@ift.ulaval.ca

H. R. Chinaei
e-mail: hchinae@gmail.com

1 Introduction

Spoken dialogue systems (SDSs) are systems which help the human user to accomplish a task using the spoken language. For example, users can use an SDS to get information about bus schedules over the phone or internet, to get information about a tourist town, to command a wheelchair to navigate in an environment, to control a music player in an automobile, to get information from customer care to troubleshoot devices, and many other tasks. Figure 1 adapted from Williams (2006) shows the architecture of an SDS. At the high level, an SDS consists of three modules: the input, the output and the control. The input includes the automatic speech recognition (ASR) and natural language understanding (NLU) components. The output includes natural language generator (NLG) and text-to-speech (TTS) components. Finally, the control module is the core part of an SDS and it consists of the dialogue model and the dialogue manager (DM).

The SDS modules work as follows. First, the ASR module receives the user utterance, i.e., a sequence of words in the form of speech signals, and makes a N-Best list containing all user utterance hypotheses. Next, NLU receives the noisy words from the ASR output, generates the possible intentions that the user could have in mind, and sends them to the control module. The control module receives the generated user intentions, possibly with a confidence score, as an observation O. The confidence score can show for instance the reliability of possible user intentions since the output generated by ASR and NLU can cause uncertainty in the machine. That is, the ASR output includes errors and the NLU output can be ambiguous, both cause uncertainty in SDS. The observation O can be used in a dialogue model to update and enhance the model.

Notice that the dialogue model and the dialogue manager interact with each other. In particular, the dialogue model

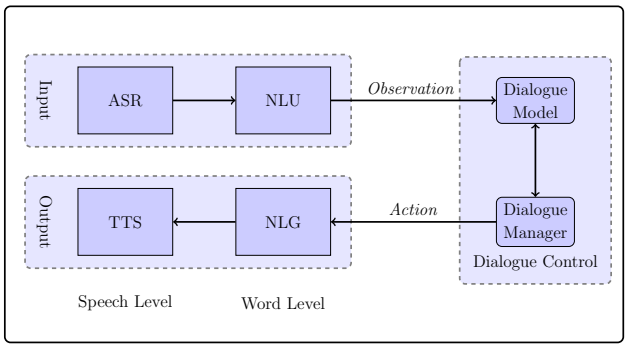

Fig. 1 The architecture of a spoken dialogue system adapted from Williams (2006)

provides the dialogue manager with the observation O and the updated model. Based on such information, the dialogue manager is responsible for making a decision. In fact, the DM updates its strategy based on the received updated model, and refers to its strategy for producing an action A, which is an input for NLG. The task of NLG is to produce a text describing the action A, and to pass the text to the TTS component. Finally, the TTS produces the spoken utterance of the text, and announces it for the user.

Building SDSs is a difficult problem since automatic speech recognition (ASR) and natural language understanding (NLU) induce errors which are the sources of uncertainty in SDSs. In addition to that, the human user behavior is not completely predictable since she may change her *intentions* during the dialogue, making thus the SDS environment stochastic.

Consider the example in Table 1 taken from SACTI-2 data set of dialogues (Weilhammer et al. 2004), where SACTI stands for simulated ASR-channel: tourist information. The first line of the table shows the user utterance, u_1. Because of the ASR errors, this utterance is misrecognized by the machine as the line in the braces shows it, i.e., $\tilde{u}_1$. The next line, m_1 reflects the machine's response to the user.

For each dialogue utterance, the machine's goal aims first to capture the user intention and then to execute the best action that satisfies the user intention. For instance, consider the ASR output in the first dialogue turn, $\tilde{u}_1$: [Is there a good restaurant week an hour tonight]. In this turn, the machine can predict the user intention as information request for *food* places since the utterance contains the only keyword *restaurant*. However, it is not the case for the ASR output in the second turn, $\tilde{u}_2$: [No I think late like uh museum price restaurant]. This utterance contains misleading words such as *museum* that can be a strong observation for other user intentions, such as user intention for *visiting* areas. Ideally, in $\tilde{u}_2$ the machine should estimate that the user is looking for *food* places, and consider *museum* as a corrupted word, since the user has been asking for *restaurant* in his previous utterance, $\tilde{u}_1$. It may however consider *museum* as a change

Table 1 A sample from the SACTI-2 dialogues (Weilhammer et al. 2004)

u_1:	Is there a good restaurant we can go to tonight
$\tilde{u}_1$:	[Is there a good restaurant week an hour tonight]
m_1 :	Would you like an expensive restaurant
u_2 :	No I think we'd like a medium priced restaurant
$\tilde{u}_2$:	[No I think late like uh museum price restaurant]
m_2 :	Cheapest restaurant is eight pounds per person
u_3 :	Can you tell me the name
$\tilde{u}_3$:	[Can you tell me the name]
m_3 :	bochka
u_4 :	Thank you can you show me on the map where it is
$\tilde{u}_4$:	[Thank you can you show me i'm there now where it is]
m_4 :	It's here
u_5 :	Thank you
$\tilde{u}_5$:	[Thank u]
u_6 :	I would like to go to the museum first
$\tilde{u}_6$:	[I would like a hour there museum first]
	…

of the user intention, and misestimate that the user is looking for *visiting* areas.

Similarly, the machine has to resolve another problem for the ASR output in the third turn, $\tilde{u}_3$: [Can you tell me the name]. Here, there is no keyword *restaurant*, however, the machine needs to estimate that the user is actually requesting information for *food* places basically because the user has been asking about *food* places in the previous utterances.

In addition, understanding the natural language is challenging. For instance, there are several ways of expressing an *intention*. This is notable for instance in SmartWheeler (see Sect. 5.2), which is an intelligent wheelchair to help persons with disabilities. SmartWheeler is equipped with an SDS, so that the users can give their commands through the spoken language besides a joystick. The users may say a command in different ways. For instance for turning right, the user may say: (i) *turn right a little please*, (ii) *turn right*, (iii) *right a little* (iv) *right*, and many other ways to express the same intention. As a response, SmartWheeler can perform the TURN RIGHT A LITTLE action or ask for REPEAT.

Such problems become more challenging when the user utterance is corrupted by ASR. For instance, SmartWheeler may need to estimate that the user asks for *turn right* from the ASR output, *10 writer little*. We call domains such as SmartWheeler *intention-based* dialogue domains. In these domains the user intention reflects the dialogue *state* which should be estimated by the machine to be able to perform the best action.

In intention-based dialogues domains, performing the best action in each dialogue state is a challenging task due to the uncertainty introduced by ASR errors and NLU problems

as well as the stochastic environment made by user behavior change. This uncertainty leads to a dialogue state which is partially observable. In domains where the decision making is sequential (as is the case in dialogues) and the state is partially observable, the suitable formal framework is the Partially Observable Markov decision process (POMDP). This framework has been extensively used to model the uncertainty and stochasticity of SDSs (Roy et al. 2000; Zhang et al. 2001a, b; Williams and Young 2007; Thomson 2009; Gašić 2011).

In POMDP, the agent does not know in which state it is in, since the state is partially observable. In consequence, it must maintain a probability distribution, known as the belief state $b(s)$, over all the possible states. At each time period, the agent has some belief state $b(s)$; it then takes an action a while observing o causing thus the environment to change from state s to state s'. In these conditions, $b(s)$ should take in consideration the new observation o and the transition from s to s' and be updated into $b(s')$ (more detail in background below). Solving a POMDP aims to maximize the expected total discounted reward over an infinite horizon and this is equivalent to find the optimal policy π^* where π is the agent's policy and it specifies an action $a = \pi(b)$ for any belief b.

Estimating the POMDP model components, as states, observations and reward is a significant issue; as the POMDP model has direct impact on the POMDP policy and consequently on the applicability of the POMDP in the domain of interest. In this context, the SDS researchers in both academia and industry have addressed several practical challenges of applying POMDPs to SDS (Roy et al. 2000; Williams 2006; Paek and Pieraccini 2008). In particular, learning the SDS dynamics ideally from the available unannotated and noisy dialogues is a challenge for us.

In many real applications including SDSs, it is usual to have large amount of unannotated data, such as web-based spoken query retrieval (Ko and Seo 2004). Otherwise, one can collect data from Wizard-of-Oz setting such as SACTI-1 (Williams and Young 2005) data, as used in this paper to learn the dialogue models. Annotating data manually, is an expensive task and consequently learning from unannotated data is an interesting challenge which is tackled using unsupervised learning methods. That is why we were interested in learning the POMDP model components based on the available unannotated data.

Furthermore, POMDPs have scalability issues. That is, finding the (near) optimal policy of the POMDP highly depends on the number of states, actions and observations. In particular, the number of observations can exponentially increase the number of conditional plans (Kaelbling et al. 1998). For example, in most non-trivial dialogue domains, the POMDP model can include hundreds or thousands of observations such as words or user utterances. In the example given in Table 1, $\tilde{u}_1$, $\tilde{u}_2$, $\tilde{u}_3$, and $\tilde{u}_4$, together with many other possible utterances, can be considered as observations. Finding the optimal policy of such a POMDP is basically intractable.

Finally, the reward model is perhaps the most handcrafted aspect of the optimization frameworks such as POMDPs (Paek and Pieraccini 2008). The reward model can be learned by observing an agent's behavior according to its optimal policy. Using inverse reinforcement learning (IRL) (Ng and Russell 2000), a reward model can be determined from behavioral observation. Fortunately, learning the reward model using IRL methods have already been proposed for the general POMDP framework (Choi and Kim 2011), paving the way for investigating its use for dialogue POMDPs.

In this paper, we propose methods for learning the dialogue POMDP model components from unannotated and noisy dialogues of intention-based dialogue domains. Our main contributions are presented in Algorithm 1 where the input to this algorithm is an unannotated dialogue set. For the experiments, we use SACTI-1 (Williams and Young 2005) and SmartWheeler dialogues (Pineau et al. 2011).

In step 1 of Algorithm 1, we learn the dialogue intentions from unannotated dialogues using an unsupervised topic modeling approach, and make use of them as the dialogue POMDP states. In step 2, we directly extract the actions from the dialogue set and learn a maximum likelihood transition model using the learned states. In step 3, we reduce observations significantly and learn the observation model. Specifically, we propose two observation models: the keyword model and the intention model. Steps 1–3 are covered in this first paper (part I).

Algorithm 1: The descriptive algorithm to learn the dialogue POMDP model components using unannotated dialogues.

Input: Given the unannotated dialogue set
Output: The learned dialogue POMDP model components that can be used in a POMDP solver to find the (near) optimal policy

1 *Learn the dialogue intentions from unannotated dialogues using an unsupervised topic modeling approach, and make use of them as the dialogue POMDP states;*
2 *Extract actions directly from dialogues and learn a maximum likelihood transition model using the learned states;*
3 *Reduce observations significantly by using only representative observations (i.e., keywords in our case) or states (i.e., intents in our case) and learn the subsequent observation model;*
4 *Learn the reward model based on the IRL technique and using the learned POMDP model components;*

Building on the learned dialogue POMDP model components, we propose in an accompanying paper (part II), two

IRL algorithms for learning the dialogue POMDP reward model from dialogues, as shown in step 4. The learned reward model makes the dialogue POMDP model complete, which can be used in an available model-based POMDP solver to find the optimal policy.

The rest of the paper is organized as follows. We describe the necessary background knowledge in Sect. 2. In Sect. 3, we cover step 1 in the descriptive Algorithm 1 by proposing learning methods for states and the transition model. Then in Sect. 4, we cover step 2 by proposing learning methods for observation model. Finally in Sect. 5, we see how we can use state and observation models in the context of two applications:SACTI-1 (Williams and Young 2005) and SmartWheeler dialogues (Pineau et al. 2011).

2 Background

As a POMDP is an extension of Markov decision process (MDP) to partial observability, let's start by a brief background on MDP. An MDP is defined as (S,A,T,γ,R,s_0) where, S is the set of discrete states, and A is the set of discrete actions. In addition, the transition model T consists of the probabilities of state transitions (dynamics of the system): $T(s,a,s') = Pr(s_{t+1} = s'|a_t = a, s_t = s)$ where s is the current state and s' is the next state. Furthermore, the reward model is $R(s,a)$ that specifies the reward of taking action a in the state s. Then γ is a number between 0 and 1 that discounts the future reward. Finally, the initial state is denoted by s_0.

In an MDP, a policy π maps each state s to an action a, i.e., $a = \pi(s)$ and the objective is to find an optimal policy π^*, where for any state s, π^* specifies an action $a = \pi^*(s)$ that *maximizes* the expected value of future rewards starting from state s:

$$V^{\pi}(s) = E_{s_t \sim T}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | \pi, s_0 = s\right] \quad (1)$$

The expected value of the reward can be computed recursively by Bellman equation defined as:

$$V^{\pi}(s) = \left[R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s')V^{\pi}(s')\right] \quad (2)$$

A POMDP is a more generalized model for planning under uncertainty. The POMDP framework considers that the states are only partially observable through some *observations*. Thus, a POMDP is represented as a tuple $(S, A, T, \gamma, R, O, \Omega, b_0\}$. Hence, a POMDP model includes an MDP model and adds O, the set of observations, Ω is the observation model defined as $\Omega(a,s',o') = Pr(o'|a,s')$ for the probability of observing o' after taking the action a which results in the state s'. Finally, the initial belief over all states is denoted by b_0.

The primary difference between POMDPs and MDPs is that, where an MDP has a particular current state, a POMDP (theoretically) maintains a probability distribution over all possible states. This distribution is called a *belief state* as stated previously. Given a belief state b, $b(s)$ is the probability assigned to a particular state s. After an action a is taken and observation o' is received, a new belief state, b', can be computed. This is done by using a *state estimator function* $SE(b,a,o')$ to calculate the probability of a new state s', $b'(s')$:

$$\begin{aligned} b'(s') &= SE(b,a,o') = Pr(s'|b,a,o') \\ &= \eta\Omega(a,s',o')\sum_{s\in S} b(s)T(s,a,s') \end{aligned} \quad (3)$$

where η is the normalization factor.

Note that an important property of the belief state is that it is a *sufficient statistic*. In other words, the belief state at time t, b_t, summarizes the initial belief b_0, as well as all the actions taken and all the observations received. Formally, we have: $b_t(s) = Pr(s|b_0, a_0, o_0, \ldots, a_{t-1}, o_{t-1})$.

In the POMDP framework, policy selects an action a for a belief state $b(s)$, i.e., $a = \pi(b(s))$. In this case, the objective in a POMDP aims to find an optimal policy π^*, where for any belief state b, π^* specifies an action $a = \pi^*(b)$ that maximizes the expected discount of future rewards starting from belief b_0:

$$V^{\pi}(b) = E_{b_t \sim SE}\left[\sum_{t=0}^{\infty} \gamma^t R(b_t, \pi(b_t)) | \pi, b_0 = b\right], \quad (4)$$

where

$$R(b,a) = \sum_{s\in S} b(s)R(s,a) \quad (5)$$

Then, we have $\pi^*(b) = \arg\max_{\pi} V^{\pi}(b)$, that is, the optimal policy. The expected value of the reward can be computed recursively using Bellman equation for V^{π}.

$$\left[V^{\pi}(b) = R(b,\pi(b)) + \gamma \sum_{o' \in O} Pr(o'|b,\pi(b))V^{\pi}(b')\right] \quad (6)$$

3 Learning states as user intentions

We consider here POMDP states as being user intentions and we learn them from dialogues using hidden topic Markov model (HTMM) (Gruber et al. 2007) as it is specified for instance in the previous work of Chinaei et al. (2009). Hidden topic Markov model, in short HTMM (Gruber et al. 2007), is an unsupervised topic modeling technique that combines latent Dirichlet allocations (LDA) (Blei et al. 2003) and HMM (Rabiner 1990) to obtain the topics of documents.

3.1 Hidden topic Markov model for dialogues

We consider the following conventions to describe HTMM for dialogues. First we assume that topics in documents are equivalent to intentions in dialogues. The intentions are hidden and the observations for the hidden intentions are words. A dialogue set $\boldsymbol{D}$ includes an arbitrary number of dialogues, $\boldsymbol{d}$, where each dialogue $\boldsymbol{d}$ consists of the recognized user utterances, $\tilde{\boldsymbol{u}}$. That is, the recognized user utterances, $\tilde{\boldsymbol{u}}$, are indeed the ASR recognition of the actual user utterance $\boldsymbol{u}$. The recognized user utterance itself, $\tilde{\boldsymbol{u}}$, is a bag of words represented as $\tilde{\boldsymbol{u}} = [w_1, \ldots, w_n]$.

In Fig. 2, a dialogue $\boldsymbol{d} \in \boldsymbol{D}$ is seen as a sequence of words w_i which are observations for a hidden intention z (i.e., a user intention). The vector $\boldsymbol{\beta}$ is a global vector that ties all the dialogues in a dialogue set $\boldsymbol{D}$, and retains the probability of words given user intentions, $Pr(w|z, \boldsymbol{\beta}) = \boldsymbol{\beta}_{wz}$. In particular, the vector $\boldsymbol{\beta}$ is drawn based on multinomial distributions with Dirichlet prior $\boldsymbol{\eta}$. On the other hand, the vector $\boldsymbol{\theta}$ is a local vector for each dialogue $\boldsymbol{d}$, and retains the probability of intentions in a dialogue, $Pr(z|\boldsymbol{\theta}) = \boldsymbol{\theta}_z$. Finally, the vector $\boldsymbol{\theta}$ is drawn from a multinomial distribution with Dirichlet prior $\boldsymbol{\alpha}$.

The parameter $\boldsymbol{\psi}_i$ aims to add the Markovian property in dialogues since successive utterances are more likely to include the same user intention. The assumption here is that a recognized utterance represents only one user intention, so all the words in the recognized utterance are observations for the same user intention. To formalize that, the HTMM algorithm assigns $\boldsymbol{\psi}_i = 1$ for the first word of an utterance, and $\boldsymbol{\psi}_i = 0$ for the rest. Then, the intention transition is possible only when $\boldsymbol{\psi}_i = 1$. Finally, the parameter ϵ is used as a prior over $\boldsymbol{\psi}$ which controls the probability of intention transition between utterances in dialogues, $Pr(z_i|z_{i-1}) = \epsilon$. Since each recognized utterance contains one user intention, we have $Pr(z_i|z_{i-1}) = 1$ for z_i, z_{i-1} within one utterance.

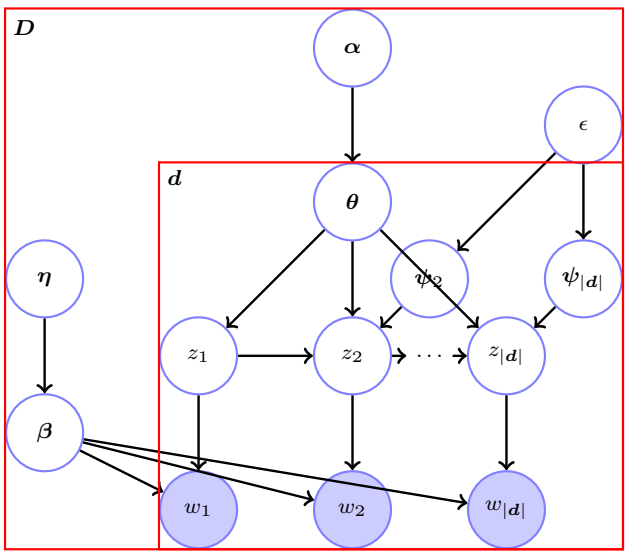

Fig. 2 The HTMM model adapted from Gruber et al. (2007), the shaded nodes are words (w) used to capture intentions (z)

Algorithm 2 describes the generative model of HTMM, adapted from Gruber et al. (2007). First, for all possible hidden intentions, the vector $\boldsymbol{\beta}$ is drawn using the multinomial distribution with prior $\boldsymbol{\eta}$. Then, for each dialogue, the vector $\boldsymbol{\theta}$ is drawn using the Dirichlet prior $\boldsymbol{\alpha}$. In line 5, for each dialogue, the vector $\boldsymbol{\theta}$ is initialized using the Dirichlet prior $\boldsymbol{\alpha}$.

The parameter $\boldsymbol{\psi}$, for each recognized utterance i in dialogue $\boldsymbol{d}$, is initialized based on a Bernoulli prior ϵ between lines 6–12. As mentioned earlier, the parameter $\boldsymbol{\psi}$ is used to add the Markovian property to the model. It determines whether the user intention for the recognized utterance i is the same as previous recognized utterance. The rest of the algorithm, lines 13–20, finds the user intentions. In line 14, if the parameter $\boldsymbol{\psi}$ is equal to 0 the algorithm assumes that the user intention for utterance i is equal to the one for utterance $i-1$, encoding thus the Markovian property. Otherwise, it draws the intention for utterance i based on the vector $\boldsymbol{\theta}$ in line 17. Recall that the vector $\boldsymbol{\theta}$ is generated using the Dirichlet distribution with prior $\boldsymbol{\alpha}$. Then in line 19 a new word w is generated based on the vector $\boldsymbol{\beta}$.

Algorithm 2: The HTMM generative model, adapted from Gruber et al. (2007).

Input: Set of dialogues $\boldsymbol{D}$, N number of intentions
Output: Generate utterances of $\boldsymbol{D}$

```
1  foreach intention z in the set of N intentions do
2  |  Draw β_z ~ Dirichlet(η);
3  end
4  foreach dialogue d in D do
5  |  Draw θ ~ Dirichlet(α);
6  |  foreach i = 1...|d| do
7  |  |  if beginning of a user utterance then
8  |  |  |  ψ_i = Bernoulli(ϵ)
9  |  |  else
10 |  |  |  ψ_i = 0
11 |  |  end
12 |  end
13 |  foreach i = 1...|d| do
14 |  |  if ψ_i = 0 then
15 |  |  |  z_i = z_{i-1}
16 |  |  else
17 |  |  |  z_i = multinomial(θ)
18 |  |  end
19 |  |  Draw w_i ~ multinomial(β_{z_i}) ;
20 |  end
21 end
```

HTMM uses Expectation Maximization (EM) and forward backward algorithm (Rabiner 1990), the standard method for approximating the parameters in HMMs. This is due to the fact that conditioned on $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, HTMM is a

special case of HMMs. In HTMM, the latent variables are user intentions z_i and $\boldsymbol{\psi}_i$ which determines if the intention for the word w_i is drawn from w_{i-1}, i.e., if $\boldsymbol{\psi}_i = 0$; or a new intention will be generated, i.e., if $\boldsymbol{\psi}_i = 1$.

Note that in HTMM, the vector $\boldsymbol{\theta}_z$ stores the probability of an intention z:

$$Pr(z|\boldsymbol{\theta}) = \boldsymbol{\theta}_z \tag{7}$$

And, the vector $\boldsymbol{\beta}_{w,z}$ stores the probability of an observation w given the intention z:

$$Pr(w|z, \boldsymbol{\beta}) = \boldsymbol{\beta}_{wz} \tag{8}$$

Learning the parameters in HTMM can be done in an "acceptable" computation time, using EM. In HTMM, the special form of the transition matrix reduces the time complexity to $O(TN)$, where T is the length of the chain for forward-backward, and N is the number of desired user intentions given to the algorithm (Gruber et al. 2007; Gruber and Popat 2007).

3.2 Learning intentions from SACTI-1 dialogues

In this section, we apply HTMM on real dialogues of the SACTI-1 domain (Williams and Young 2005).[1] It contains 144 dialogues between 36 users and 12 experts who play the role of the machine for 24 total tasks on this data set. The utterances are first recognized using a speech recognition error simulator, and then sent to human experts for a response. There are four levels of ASR noise in SACTI-1 data: *none*, *low*, *medium*, and *high* noise. Finally, 2048 utterances, with 817 distinct words, are used in our experiments.

Table 2 shows a dialogue sample from SACTI-1. The first line of the table shows the first user utterance, u_1. Because of ASR errors, this utterance is recognized as $\tilde{u}_1$. Then, m_1 is the actual machine utterance representing a response to the user request recognized by the ASR in $\tilde{u}_1$.

We applied HTMM as introduced in the previous section to learn possible user intentions in SACTI-1. For that, we removed the machine responses from the dialogues in order to learn the user intentions by only considering the noisy user utterances. The parameter N for the number of intentions were assigned based on the application experiments. That is, we varied the number of intentions from 1–10 and learned the whole POMDP. Then the POMDP model that received highest average reward in simulation runs were selected. If the average rewards were closed we picked the lower N, since lower number of intents tend to capture more meaningful intents.

Table 3 shows the learned intentions from SACTI-1 data, using HTMM. The algorithm learns three user intentions.

[1] see http://mi.eng.cam.ac.uk/projects/sacti/corpora/.

Table 2 A sample from the SACTI-1 dialogues (Williams and Young 2005)

	…
u_1	yeah hello this is johan schmulka uh and i'm uh searching for a bar
	in this town can you may be tell me where the cafe blu is
$\tilde{u}_1$	[hello this is now seven four bus and do you tell me where to cafe blu is]
m_1	cafe blu is on alexander street
u_2	oh um yeah how can i get to alexander street and
	where exactly is it i know there a shopping area on alexander street um
$\tilde{u}_2$	[i am yeah i am at the alexander street and
	where is it was on a the center of alexander street]
m_2	it is on the east side of alexander street so
	%um it's %um just off middle road
	…

By looking at the top words of each intent we can name intents as follows : *visits*, *transports*, and *foods*, respectively. Each intention is represented by its 20-top words with their probabilities. In Table 3, we have highlighted only the words which best represent each intention; these highlighted words are called *keywords*. To extract these keywords, we avoided stop words such as *the, a, an, to*. For instance, the words *hotel*, *tower*, and *castle* are keywords which represent the user intentions for the required information about visiting areas, i.e., *visits*.

Then, for each recognized user utterance $\tilde{u} = [w_1, \ldots, w_n]$, we define its subsequent state as the highest probable intention z:

$$s = \underset{z}{\text{argmax}}\, Pr(w_1, \ldots|z) = \underset{z}{\text{argmax}} \prod_i Pr(w_i|z) \tag{9}$$

where $Pr(w_i|z)$ is already learned and stored in the parameter $\boldsymbol{\beta}_{wz}$ according to Eq. (8). The second equality in the equation, the product of probabilities, is due to the independence of words given an user intention.

User intentions have been previously suggested to be used as states of dialogue POMDPs (Roy et al. 2000; Zhang et al. 2001b; Matsubara et al. 2002; Doshi and Roy 2007, 2008). However, to the best of our knowledge, they have not been automatically extracted from real data. Here, we learn the user intentions based on unsupervised learning methods. This enables us to use raw data, with little annotation or preprocessing.

In a previous work Chinaei et al. (2009), we were able to learn 10 user intentions from SACTI-2 dialogues (Weilhammer et al. 2004), without annotating data or any prepossessing. In this paper, we were able to estimate the user intentions behind utterances when users did not use a keyword for an

Table 3 The learned user intentions from the SACTI-1 dialogues. We have highlighted in bold only the words which best represent each intention

intention 1		*visits*	
the	0.08	like	0.01
i	0.06	**hotel**	0.01
to	0.05	for	0.01
um	0.02	would	0.01
is	0.02	i'm	0.01
a	0.02	**tower**	0.01
and	0.02	**castle**	0.01
you	0.02	go	0.01
uh	0.02	do	0.01
what	0.01	me	0.01
intention 2		*transports*	
the	0.08	a	0.02
to	0.04	does	0.02
is	0.04	**road**	0.02
how	0.03	and	0.01
um	0.02	on	0.01
it	0.02	long	0.01
uh	0.02	of	0.01
i	0.02	much	0.01
from	0.02	**bus**	0.01
street	0.02	there	0.01
intention 3		*foods*	
you	0.06	um	0.02
the	0.04	and	0.2
i	0.04	thank	0.01
a	0.03	to	0.01
me	0.03	of	0.01
is	0.02	**restaurant**	0.01
uh	0.02	there	0.01
can	0.02	do	0.01
tell	0.02	could	0.01
please	0.02	where	0.01

intention. In addition, we were able to learn the true intention behind recognized utterances that included wrong keywords or multiple keywords.

3.3 Learning the transition model

In the previous section, we learned states of the dialogue POMDP. In this section, we go through the second step of our descriptive Algorithm 1: extracting actions directly from dialogues and learning a maximum likelihood transition model.

Table 4 Learned probabilities of intentions for the recognized utterances in the SACTI-1 example

	…
u_1	yeah hello this is johan schmulka uh and i'm uh searching for a bar
	in this town can you may be tell me where the cafe blu is
$\tilde{u}_1$	[hello this is now seven four bus and do you tell me where to cafe blu is]
p_1	*t:0.000000 v:0.000000 f:1.000000*
u_2	oh um yeah how can i get to alexander street and
	where exactly is it i know there a shopping area on alexander street um
$\tilde{u}_2$	[i am yeah i am at the alexander street and
	where is it was on a the center of alexander street]
p_2	*t:0.999992 v:0.000008 f:0.000000*
	…

In Sect. 2, we saw that a transition model is in the form of $T(s, a, s')$ where T stores the probability of going to the state s' when we perform the action a in the state s. We learn a maximum likelihood transition model by performing the following counting:

$$T(s, a, s') = Pr(s'|s, a) \propto \frac{Count(s, a, s')}{Count(s, a)} \tag{10}$$

To do so, we extract the set of possible actions from the dialogue set. Then, the maximum probable intention (state) is assigned to each recognized utterance using Eq. (9). For instance, for the recognized utterances in the SACTI-1 example, we can learn the probability distribution of the intentions from Eq. (8), denoted by p in Table 4. Then, to calculate the state for each recognized utterance, we take the maximum probable state, using Eq. (9). For instance, the user intention for u_2 is learned as t, i.e., *transports*. Finally, the transition model is learned using Eq. 10.

Notice that not every possible triple (s, a, s') does occur in the data, so some of the probabilities in Eq. (10) could be zero. We avoid this by adding one to the numerator in Eq. (10), a technique known as *smoothing*. In Eq. (11) we add 1, as many as count of (s, a, s'), in the numerator, so we should add $Count(s, a, s')$ to the denominator so that the equation sums to one. Therefore, the transition model can be calculated as:

$$\begin{aligned} T(s, a, s') &= Pr(s'|s, a) \\ &= \frac{Count(s, a, s') + 1}{Count(s, a) + Count(s, a, s')} \end{aligned} \tag{11}$$

The transition model introduced in Eq. (11) is similar to the user goal model for the factored transition model proposed by Williams and Young (2007), Williams (2006).

4 Learning observations and observation models

We go now through the third step in the descriptive Algorithm 1. That is, reducing the observations significantly and learning the observation model. However, the definition of observations and observation model can be non-trivial. In particular, the time complexity for learning the optimal strategy of a POMDP is double exponential in the number of observations (Pineau et al. 2003). In non-trivial domains such as dialogue manager (DM), the number of observations is large. Depending on the domain, there can be hundreds or thousands of words which ideally should be used as observations. In this case, solving a POMDP with such a huge number of observations is intractable. In order to be able to solve these POMDPs, we need to reduce the number of observations significantly.

4.1 Keyword observation model

For each state, this model uses the 1-top keyword which best represents the state. For instance, for SACTI-1 dialogues the 1-top keyword in Table 3 are the observations which include *hotel, road*, and *restaurant*. These observations can best represent the states: *visits, transports*, and *foods*, respectively. In addition, an auxiliary observation, which is called *confusedObservation*, is used, when none of the keyword observations occurs in a recognized user utterance. If an utterance includes more than one of the keyword observation, the *confusedObservation* is also used as observation.

For the keyword observation model, we define a maximum likelihood observation model:

$$\Omega(o', a, s') = Pr(o'|a, s') \propto \frac{Count(a, s', o')}{Count(a, s')}$$

To make a more robust observation model, we apply smoothing to this maximum likelihood observation model similarly to what we did for the state-model in Eq. 11:

$$\Omega(o', a, s') = Pr(o'|a, s') = \frac{Count(a, s', o') + ss1}{Count(a, s') + Count(a, s', o')}$$

In the experiment of the observation models, in Sect. 5.2.2, the dialogue POMDP with the keyword observation model is called *keyword POMDP*.

4.2 Intention observation model

Given the recognized user utterance $\tilde{u} = [w_1, \ldots, w_n]$, the observation o is defined in the same way as the state, i.e., the highest probable underlying intention in Eq. (9). So the observation o would be:

$$o = \underset{z}{\operatorname{argmax}} \prod_i Pr(w_i|z) \tag{12}$$

Recall that $Pr(w_i|z)$ is learned and stored in the vector $\boldsymbol{\beta}_{w_i z}$ from Eq. (8).

Notice that for the intention model, each state itself is the observation. As such, the set of observation is equivalent to the set of states. For instance, for SACTI-1 example the intention observations are *vo, to*, and *fo* respectively for *visits, transports*, and *foods* states.

In the intention observation model, we essentially end up with an MDP model. This is because we use the highest probable intention as state and we use the highest probable intention as observation as well. So, we end up with a deterministic observation model, which is such as an MDP, as discussed in Sect. 2. However, we can use a sort of smoothing to allow a small probability for other observations than the observation corresponding to the current state. In the experiment of the observation models, Sect. 5.2.2, we use the intention model without smoothing as the learned intention MDP model.

For the intention observation model, we can estimate it using the recognized utterances $\tilde{u}$ inside the training dialogue $\boldsymbol{d}$, and using the vector $\boldsymbol{\beta}_{wz}$ and $\boldsymbol{\theta}_z$, reflected in Eqs. (8) and (7), respectively. In the experiment of the observation models, Sect. 5.2.2, the dialogue POMDP with the intention observation model is called *intention POMDP*.

5 Experiments

5.1 SACTI dialogues

We now use the proposed methods in previous sections to learn a dialogue POMDP from SACTI-1 dialogues. First, we use the learned intentions in Table 3 as states of the domain. Based on the captured intentions, we defined 3 primary states for the SACTI-1 machine as follows: *visits (v)* , *transports (t)* , and *foods (f)*. We define two terminal states, *success* and *failure* for dialogues that end successfully and unsuccessfully (respectively). The notion of successful or unsuccessful dialogue is defined by user. After finishing each dialogue, the user assigns the level of precision and recall of the received information, after finishing each dialogue. This is the only explicit feedback that we require to define the terminal states of dialogue POMDP. A dialogue is successful if its precision and recall are above a predefined threshold.

The set of actions comes directly from SACTI-1 dialogue set, and they include: *Inform*, *Request*, *GreetingFarewell*, *ReqRepeat*, *StateInterp*, *IncompleteUnknown*, *ReqAck*, *ExplAck*, *HoldFloor*, *UnsolicitedAffirm*, *RespondAffirm*, *RespondNegate*, *RejectOther*, *DisAck*. For instance, *GreetingFarewell* is used for initiating or ending a dialogue, *Inform* is used for giving information for a user intention, *ReqAck* is used for the machine request for user acknowledgement; *StateInterp* is used for interpreting the intentions of user.

Using such states and actions, the transition model of our dialogue POMDP was learned based on the method in Sect. 3.3.

The observations for SACTI-1 would be *hotel, street, restaurant, confusedObservation, success, failure* in the case of keyword observation model, and the observations would be *vo, to, fo, success, failure* in the case of intention observation model. Then, based on the proposed methods presented in the previous section, both keyword and intention observation models are learned.

For our experiments, we used a typical reward model. Similar to previous works, we penalized each dialogue turn by −1 (Williams and Young 2007). Moreover, actions in the success (terminal) state get +50 as reward and actions in the failure (terminal) state get -50 reward.

Table 5 represents the sample described in Table 2, after applying the two observation models on the dialogues. The first user utterance is shown in u_1. Note that u_1 is hidden to the machine and is recognized as the line in $\tilde{u}_1$. Then, $\tilde{u}_1$ is reduced and received as the observation in o_1; if the keyword observation model is used the observation will be *confusedObservation*. This is because none of the keywords *hotel, street*, and *restaurant* occur in $\tilde{u}_1$. But, if the intention observation model is used then the observation inside parenthesis is used, i.e., *fo* which is an observation with high probability for *foods* state, and with small probability for *visits* and *transports* states.

The next line, a_1 shows the machine action in the form of dialogue acts. For instance, *Inform(foods)* is the machine dialogue act which is uttered by the machine as m_1, i.e., *cafe blu is on alexander street*. Next, the table shows u_2, $\tilde{u}_2$, o_2, and a_2. Note that in o_2, as opposed to o_1 in the case of keyword observation model, the keyword *street* occurs in the recognized utterance $\tilde{u}_2$.

Table 5 Results of applying the two observation models on the SACTI-1 sample

	…
u_1	yeah hello this is johan schmulka uh and
	i'm uh searching for a bar in this town
	can you may be tell me where the cafe blu is
$\tilde{u}_1$	[hello this is now seven four bus
	and do you tell me where to cafe blu is]
o_1	*confusedObservation (fo)*
a_1:	*Inform(foods)*
m_1	cafe blu is on alexander street
u_2	oh um yeah how can i get to alexander street and
	where exactly is it i know there a shopping area
	on alexander street um
$\tilde{u}_2$	[i am yeah i am at the alexander street and
	where is it was on a the center of alexander street]
o_2	*street (to)*
a_2:	*Inform(transports)*
m_2	it is on the east side of alexander street so
	%um it's %um just off middle road
	…

5.1.1 HTMM evaluation

We evaluated HTMM for learning user intentions in dialogues. To achieve that, we measured the performance of the model on the SACTI data set based on the definition of *perplexity* similar to Blei et al. (2003), Gruber et al. (2007). For a learned topic model on a train data set, perplexity can be considered as a measure of on average how many different equally most probable words can follow any given word. Therefore, it measures how difficult it is to estimate the words from the model. So, the lower the perplexity is, the better is the model.

Formally, the perplexity of a test dialogue $\boldsymbol{d}$ after observing the first k words can be drawn using the following equation:

$$Perplexity = \exp\left(-\frac{\log\ Pr(w_{k+1},\ldots,w_{|\boldsymbol{d}|}|w_1,\ldots,w_k)}{|\boldsymbol{d}|-k}\right)$$

We can manipulate the probability distribution in the equation above as:

$$Pr(w_{k+1},\ldots,w_{|\boldsymbol{d}|}|w_1,\ldots,w_k) = \sum_{i}^{N} Pr(w_{k+1},\ldots,w_{|\boldsymbol{d}|}|z_i)Pr(z_i|w_1,\ldots,w_k)$$

where z_i is a user intention in the set of N captured user intentions from the train set. Given a user intention z_i, probability of observing $w_{k+1},\ldots,w_{|\boldsymbol{d}|}$ are independent of each other, so we have:

$$Pr(w_{k+1},\ldots,w_{|\boldsymbol{d}|}|w_1,\ldots,w_k) = \sum_{i}^{N}\prod_{j=k+1}^{|\boldsymbol{d}|} Pr(w_j|z_i)Pr(z_i|w_1,\ldots,w_k)$$

To find out the perplexity, we learned the intentions for each test dialogue $\boldsymbol{d}$ based on the first k observed words in $\boldsymbol{d}$, i.e., $\boldsymbol{\theta}_{new} = Pr(z_i|w_1,\ldots,w_k)$ is calculated for each test dialogue, whereas the vector $\boldsymbol{\beta}$, which retains $Pr(w_j|z_i)$ (cf. Eq. (8)), is learned from the training dialogues. We calculated the perplexity for 5 % of the dialogues in data set and we used the 95 % rest for training. Figure 3 shows the average perplexity after observing the first k utterances of test dialogues. As the figure shows, the perplexity is reduced significantly when we observe new utterances.

We already mentioned that HTMM has an "acceptable" computation time since it has a special form of the transition

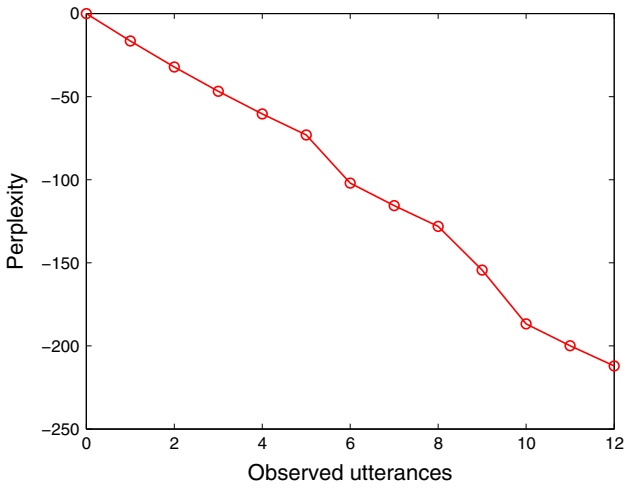

Fig. 3 Perplexity trend with respect to increase of number of observed user utterances

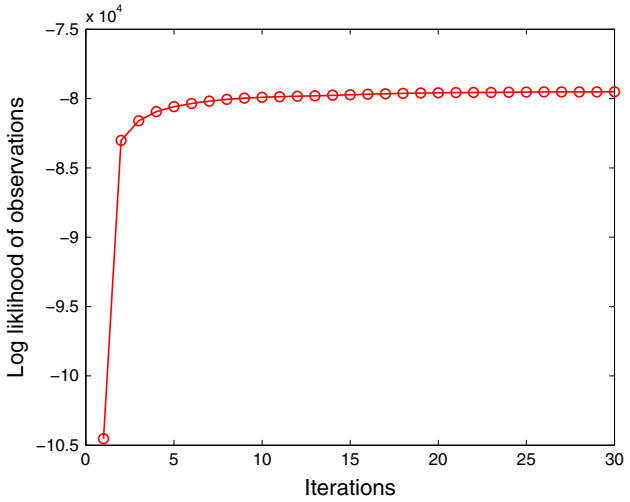

Fig. 4 Log likelihood of observations in HTMM as a function of number of iterations

matrix (Gruber et al. 2007; Gruber and Popat 2007). Here we show that the convergence rate of HTMM based on the convergence of log likelihood of data. Figure 4 shows the log likelihood of the observations for 30 iterations of the algorithm. We can see in the figure that the algorithm converges quite fast. For the given observations, the log likelihood is computed by averaging over possible intentions, we have:

$$\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = \sum_{i=1}^{|\boldsymbol{D}|} \sum_{j=1}^{|\boldsymbol{d}_i|} \log \sum_{t=1}^{N} Pr(w_{i,j} = w | z_{i,j} = z_t)$$

5.1.2 Learned POMDP evaluation

We evaluated the learned intention POMDP from SACTI-1 dialogues, introduced in Sect. 3.2, using simulation runs. The

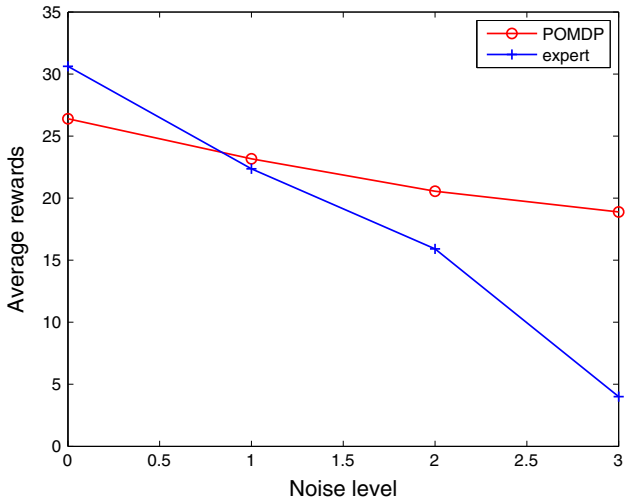

Fig. 5 Average rewards accumulated by the learned dialogue POMDPs with respect to the noise level. The x-axis represents the noise level: 0, 1, 2, and 3 are respectively for no noise, low noise, medium noise, and high noise

learned intention dialogue POMDP models from SACTI-1 consist of 3 primary states and 2 terminal states, 14 actions, and 5 intention observations. We solved our POMDP models, using the ZMDP software (with HSVI2 algorithm) available online.[2] We set a uniform distribution on the 3 primary states, *visits, transports*, and *foods*, and set the discount factor to 0.90.

Based on simulation runs, we evaluate the robustness of our learned POMDP models. For SACTI data, noise in ASR has four levels, from 0 to 3 for none, low, medium, and high levels (respectively). For each noise level, we randomly take 24 available expert dialogues, then calculate average accumulated rewards for the experts from the subsequent 24 expert dialogues, and made a dialogue POMDP model from these subsequent 24 expert dialogues. Then, for each POMDP we performed 24 simulations and calculate average accumulated rewards produced by these simulations. In our experiments, we used the default simulation in the ZMDP software.

Figure 5 plots the average accumulated rewards as the noise level changes from 0 to 3 for none, low, medium, and high levels of noise (respectively). As the figure shows, the dialogue POMDP models are robust to the ASR noise levels. That is, performance of the learned dialogue POMDPs decreases only slightly as the noise level increases, whereas performance of experts decreases significantly, in particular at high level of noise. Notice that in Fig. 5 the average accumulated mean reward for the experts is the highest when there is no noise, and it is higher than the subsequent learned POMDPs. This is reasonable as the human expert can have

[2] at: http://www.cs.cmu.edu/~trey/zmdp/.

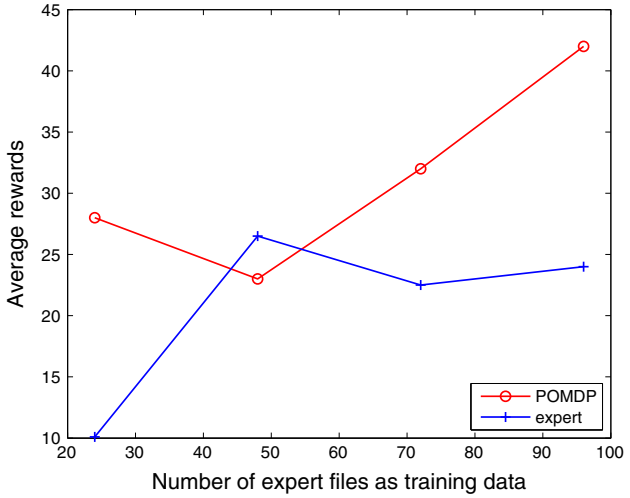

Fig. 6 Average rewards accumulated by the learned dialogue POMDPs with respect to the size of expert dialogues as training data

best performance in the least uncertain conditions, i.e., when there is no noise.

We evaluate the performance of the learned dialogue POMDPs as a function of expert dialogues (as training data), shown in Fig. 6. Similar to the previous experiments, we calculated the average accumulated rewards for the learned POMDPs and for the experts from the subsequent expert dialogues. Overall, performance of the learned dialogue POMDPs is directly related to the number of expert files and we find that more training data implies better performance.

Table 6 shows a sample from the learned dialogue POMDP simulation. The first action, a_1, is generated by dialogue POMDP, which is shown in the form of natural language in the following line, denoted by m_1. Then, the observation o_2 is generated by environment, *vo*. For instance, the recognized user utterance could have been an utterance such as: $\tilde{u}$: *I would like a hour there museum first*, and therefore its intention observation can be calculated using Eq. (12).[3] Then, based on the received observation the POMDP belief, shown in b_1, is updated, using Eq. (3). Based on belief b_1, the dialogue POMDP performs the next action, denoted by a_2.

In turns 3–5 shown in Table 6, we can see that the learned dialogue POMDP performs intuitively. In turn 3, the dialogue POMDP informs the user about *transports*, after receiving the observation *to* in turn 2 (the observation for *transports*). In a_4, the dialogue POMDP requests for acknowledgement that the user actually looks for *transports*, perhaps since it has already informed the user about *transports* in turn 3. After receiving the observation *to* in turn 4, and updating the belief, the dialogue POMDP informs the user again about *transports* in a_5.

[3] Notice that these results are only based on the dialogue POMDP simulation; where there exists neither user utterance nor machine's utterance but only the simulated action and observations.

Table 6 A sample from SACTI-1 dialogue POMDP simulation

a_1:	*GreetingFarewell*
m_1:	How can I help you?
o_2:	*vo*
b_1:	*t:0.04814 v:0.91276 f:0.03909*
a_2:	*Inform(visits)*
m_2:	Here is information about visiting areas
o_2:	*to*
b_2:	*t:0.96732 v:0.00818 f:0.02449*
a_3:	*Inform(transports)*
m_3:	*Here is information about transportation*
o_3:	*to*
b_3:	*t:0.99385 v:0.00031 f:0.00583*
a_4:	*ReqAck(transports)*
m_4:	*Are you looking for transportation*
o_4:	*to*
b_4:	*t:0.94565 v:0.04833 f:0.00600*
a_5:	*Inform(transports)*

5.2 Dialogue POMDP model learning for SmartWheeler

We learned the possible user intentions in SmartWheeler dialogue based on the HTMM method as explained in Sect. 3.1. To do so, we preprocessed the dialogues to remove stop words such as determiners and auxiliary verbs. Then, we learned the user intentions for the SmartWheeler dialogues. Table 7 shows the learned user intentions with their four top words. Most of the learned intentions show a specific user *command*:

i_1: *move forward little*, i_2: *move backward little*,
i_3: *turn right little*, i_4: *turn left little*,
i_5: *follow left wall*, i_6: *follow right wall*,
i_8: *go door*, and i_{11}: *stop*.

There are two learned intentions that loosely represent a command:

i_9: *set speed* and i_{10}: *follow person*.

Finally, there is a learned intention that represent two commands:

i_7: *turn degree right/left*.

Table 8 shows results of HTMM application on SmartWheeler for the example shown in Table 8. The line denoted

Table 7 The learned user intentions from the SmartWheeler dialogues. We have highlighted in bold only the words which best represent each intention

intention	1	*intention*	5	*intention*	9
forward	0.180	**left**	0.242	**for**	0.088
move	0.161	wall	0.229	word	0.080
little	0.114	follow	0.188	speed	0.058
drive	0.081	fall	0.032	set	0.054
intention	2	intention	6	*intention*	10
backward	0.380	**right**	0.279	**top**	0.143
drive	0.333	wall	0.212	stop	0.131
little	0.109	follow	0.197	follow	0.098
top	0.017	left	0.064	person	0.096
intention	3	*intention*	7	*intention*	11
right	0.209	**turn**	0.373	**stop**	0.942
turn	0.171	degree	0.186	stopp	0.022
little	0.131	right	0.165	scott	0.007
bit	0.074	left	0.162	but	0.002
intention	4	*intention*	8		
left	0.189	**go**	0.358		
turn	0.171	door	0.289		
little	0.138	forward	0.071		
right	0.090	backward	0.065		

by u is the true user utterance, manually extracted by listening to the dialogue recordings. Then, $\tilde{u}$ is the recognized user utterance by ASR. For each recognized utterance, the following three lines show the probability of each user intention, denoted by p. Finally, the last line, denoted by a, shows the performed action by SmartWheeler.

For instance, the second utterance u_2 shows that the user actually uttered *turn right a little*, but it is recognized as *10 writer little* by ASR. The most probable intention returned by HTMM for this utterance is i_3: *turn right little* with 0.99 probability. This is because HTMM considers Markovian property for deriving intentions, cf. Sect. 3.1. Consequently, in the second turn the intention i_3 gets high probability since in the first turn the user intention is i_3 with high probability.

Before we learn a complete dialogue POMDP, first we learned an intention MDP using the SmartWheeler dialogues. We used the learned intentions, $i_1, \ldots, i_{11}$, as the states of the MDP. The learned states are presented in Table 9. Note that for the intention i_7, we used it as the state for the command *turn degree right* as in the intention i_7 the word *right* occurs with slightly higher probability than the word *left*.

Then, we learned the transition model, i.e., the smoothed ML transition method, introduced in Sect. 3.3. Note that the dialogue MDP here is in fact an intention MDP in the same way defined in Sect. 4. That is, we used a deterministic intention observation model for the dialogue MDP, which considers the observed intention as its current state during the dialogue interaction.

5.2.1 Observation model learning

Built off the learned dialogue MDP, we developed two dialogue POMDPs by learning the two observation sets and their subsequent observation models: keyword model and intention model as proposed in Sect. 4. From these models, we then developed the keyword dialogue POMDP and the intention dialogue POMDP for SmartWheeler. As mentioned in Sect. 5.1.2, we first show the two observation sets for SmartWheeler and then compare the intention POMDP performance to the keyword POMDP performance.

The keyword observation model for each state uses a keyword that best represents the state. We use the *1-top* word of each state, shown in Table 7, as observations (the highlighted words). In this case, the observations are:

forward, backward, right, left, turn, go, for, top, stop.

Note that states s_3 and s_6 share the same keyword observation, i.e. *right*. Also, states s_4 and s_5 share the same keyword observation, i.e., *left*.

For the intention model, each state itself is the observation. Then, the set of observations is equivalent to the set of intentions. For SmartWheeler the intention observations are:

$$i_1o,\ i_2o,\ i_3o,\ i_4o,\ i_5o,\ i_6o,\ i_7o,\ i_8o,\ i_9o,\ i_{10}o,\ i_{11}o.$$

respectively for the states:

$$s_1,\ s_2,\ s_3,\ s_4,\ s_5,\ s_6,\ s_7,\ s_8,\ s_9,\ s_{10},\ s_{11}.$$

Table 8 A sample from the results of applying HTMM on SmartWheeler

	...
u_1 :	turn right
$\tilde{u}_1$:	[turn right]
	i_1 : 7.1e-9 i_2 : 9.6e-10 i_3 : **0.6**
p_1	i_4 : 0.2 i_5 : 2.6e-8 i_6 : 2.2e-5
	i_7 : 0.1 i_8 : 6.3e-7 i_9 : 1.6e-8 i_{10} : 2.4e-6 i_{11} : 5.2e-9
a_1 :	TURN RIGHT
u_2 :	turn right a little
$\tilde{u}_2$:	[10 writer little]
	i_1 : 0.0 i_2 : 0.0 i_3 : **0.9**
p_2	i_4 : 0.0 i_5 : 1.3e-7 i_6 : 5.8e-8
	i_7 : 8.8e-8 i_8 : 1.2e-6 i_9 : 5.9e-5 i_{10} : 8.8e-5 i_{11} : 1.1e-7
a_2 :	PLEASE REPEAT YOUR COMMAND
u_3 :	turn right a little
$\tilde{u}_3$:	[turn right to lead a]
	i_1 : 6.1e-11 i_2 : 9.5e-12 i_3 : **0.9**
p_3	i_4 : 0.0 i_5 : 2.7e-08 i_6 : 2.0e-07
	i_7 : 0.0 i_8 : 3.9e-9 i_9 : 1.9e-10 i_{10} : 4.4e-08 i_{11} : 1.7e-11
a_3 :	TURN RIGHT A LITTLE
u_4 :	stop
$\tilde{u}_4$:	[stop]
	i_1 : 3.2e-5 i_2 : 4.8e-6 i_3 : 0.0
p_4	i_4 : 0.0 i_5 : 0.0 i_6 : 7.8e-6
	i_7 : 0.0 i_8 : 0.0 i_9 : 0.0 i_{10} : 0.0 i_{11} : **0.9**
a_4 :	STOP
	...

Table 9 The SmartWheeler learned states

s_1	*move-forward-little*
s_2	*move-backward-little*
s_3	*turn-right-little*
s_4	*turn-left-little*
s_5	*follow-left-wall*
s_6	*follow-right-wall*
s_7	*turn-degree-right*
s_8	*go-door*
s_9	*set-speed*
s_{10}	*follow-person*
s_{11}	*stop*

Table 10 shows the sample dialogue from SmartWheeler after learning the two observation sets. In this table, line o_1 is the observation for the recognized utterance by ASR, $\tilde{u}_1$. If the keyword observation model is used the observation will be *right*, however, if intention observation model is used then the observation will be the one inside parenthesis, i.e., i_3o. In fact, i_3o is an observation with high probability for the state s_3, and with low probability for the rest of states.

Note that in o_2 for the case of keyword observation, the observation is *confusedObservation*. This is because for the keyword model, none of the keyword observations occurs in the recognized utterance $\tilde{u}_2$. However, the intention observation interestingly becomes i_3o which is the same as the intention observation in o_1.

5.2.2 Comparison of intention POMDP and keyword POMDP

Recall from the previous section that in the keyword POMDP, the observation set is the set of learned keywords and the observation model is the learned keyword observation model. In the intention POMDP, however, the observation set is the set of learned intentions and the observation model is the learned intention observation model. The learned keyword and intention POMDPs are then compared based on their

Table 10 A sample from the results of applying the two observation models on the SmartWheeler dialogues

	...
u_1 :	turn right
$\tilde{u}_1$:	[turn right]
o_1 :	*right* ($i_3 o$)
u_2 :	turn right a little
$\tilde{u}_2$:	[10 writer little]
o_2 :	*confusedObservation* ($i_3 o$)
u_3 :	turn right a little
$\tilde{u}_3$:	[turn right to lead a]
o_3 :	*right* ($i_3 o$)
u_4 :	stop
$\tilde{u}_4$:	[stop]
o_4 :	*stop* ($i_{11} o$)
	...

policies. To do so, we assumed a reward model for the two dialogue POMDPs and compared the optimal policies of the two POMDPs, based on their accumulated mean rewards in simulation runs.

Similar to the previous work of Png and Pineau (2011), we considered reward of +1 for the SmartWheeler performing the right action at each state, and 0 otherwise. Moreover, for the general query, PLEASE REPEAT YOUR COMMAND (as shown in Table 10), the reward is considered as +0.4 for each state where this query occurs. The intuition for this reward is that in each state it is best to perform the right action of the state, and it is better to perform a general query action than to perform any other wrong action in the state. That is the reason for defining the +0.4 reward for the query action ($0 < +0.4 < 1$).

The dialogue POMDP models consist of 11 states, 12 actions and 10 observations if the keyword observation model is used (9 keywords and the *ConfusedObservation*). Otherwise, there are 11 observations for the intention observation model. We set a uniform distribution on states, and set the discount factor to 0.9.

Similar to Sect. 5.1.2, we evaluated our learned observation models based on accumulated mean rewards. This is because the reward model is the same for the intention POMDP and keyword POMDP. Then, the learned policy of each model can reflect the quality of the learned observation model.

We used the default simulation in ZMDP software which simulates the environment by randomly sampling observations and uses the provided observation and transition models. Note that since the transition model is the same for the intention POMDP and keyword POMDP, the accumulated reward by policy of each model can demonstrate the quality of the observation model.

Table 11 The performance of the intention POMDP vs. the keyword POMDP, learned from the SmartWheeler dialogues

	Mean reward	*Conf95Min*	*Conf95Max*
Int. POMDP	8.914	8.904	8.922
Key. POMDP	4.784	4.767	4.802

Table 11 shows the comparison of the two models based on 1,000 simulation runs. The table shows that the intention POMDP accumulates strongly higher mean reward than the keyword POMDP based on 1000 simulation runs by ZMDP software. In Table 11, *Conf95Min* and *Conf95Max* are respectively the minimum 95 % confidence and the maximum 95 % confidence of the accumulated mean reward. This means that with approximately 95 % confidence the accumulated mean reward occurs inside the interval formed by *Conf95Min* and *Conf95Max*.

6 Conclusion and future work

We introduced machine learning techniques to learn the set of possible user intentions from dialogues and used them as POMDP states. We also learned the associated POMDP transition model from data through a maximum likelihood model. To achieve that, we used HTMM to learn the dialogue POMDP intentions, mainly because HTMM considers the Markovian property inside dialogues and it is computationally efficient.

Since it is crucial to reduce the observation state size, we then proposed two observation models: the keyword model and the intention model. Using these two models, the number of observations was reduced significantly while the POMDP performance remains high particularly in the intention POMDP.

Our first experiments on SACTI (tourist information) and SmartWheeler (a dialogue between users and an intelligent wheelchair) are promising since they demonstrated good performance. Certainly, the unsupervised approach using dialogues from experts for learning POMDP components, that we have defended in this paper, opens the door to future work.

One possible direction for future work can be application of other topic modeling approaches such as the LDA (Blei et al. 2003). A survey of topic modeling methods can be found in Blei (2012), Daud et al. (2010).

Another possible direction concerns the proposed learned observation models which could be further extended and enhanced for instance by merging the keyword observations and intention observations. Furthermore, other methods could be used for learning the observation model such as Bayesian-based methods (Atrash and Pineau 2010; Doshi and Roy 2008; Png and Pineau 2011; Png et al. 2012). In par-

ticular, Png and Pineau (2011), Png et al. (2012) proposed an online Bayesian approach for updating the observation model which can be extended for learning the observation model of dialogue POMDPs from SmartWheeler dialogues.

References

Atrash, A., & Pineau, J. (2010). A Bayesian method for learning POMDP observation parameters for robot interaction management systems. In *The POMDP practitioners workshop*.

Blei, D. (2012). Introduction to probabilistic topic models. *Communications of the ACM, 55*(4), 77–84.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Chinaei, H. R., Chaib-draa, B., & Lamontagne, L. (2009). Learning user intentions in spoken dialogue systems. In *Proceedings of the 1st International Conference on Agents and Artificial Intelligence (ICAART'09)*, Porto, Portugal.

Choi, J., & Kim, K.-E. (2011). Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, *12*, 691–730.

Daud, A., Li, J., Zhou, L., & Muhammad, F. (2010). Knowledge discovery through directed probabilistic topic models: A survey. *Frontiers of Computer Science in China*, *4*(2), 280–301.

Doshi, F., & Roy, N. (2007). Efficient model learning for dialog management. In *Proceedings of the 2nd ACM SIGCHI/SIGART conference on Human-Robot Interaction (HRI'07)*, Arlington, Virginia, USA.

Doshi, F., & Roy, N. (2008). Spoken language interaction with model uncertainty: An adaptive human-robot interaction system. *Connection Science*, *20*(4), 299–318.

Gašić, M. (2011). Statistical dialogue modelling. *PhD thesis*, Department of Engineering, University of Cambridge.

Gruber, A., & Popat, A. (2007). Notes regarding computations in open htmm. http://openhtmm.googlecode.com/files/htmm_computations.pdf

Gruber, A., Rosen-Zvi, M., & Weiss, Y. (2007). Hidden topic Markov models. In *Artificial intelligence and statistics (AISTATS'07)*, San Juan, Puerto Rico, USA.

Kaelbling, L., Littman, M., & Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1–2), 99–134.

Ko, Y., & Seo, J. (2004). Learning with unlabeled data for text categorization using bootstrapping and feature projection techniques. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL'04)*, Barcelona, Spain.

Matsubara, S., Kimura, S., Kawaguchi, N., Yamaguchi, Y., & Inagaki, Y. (2002). Example-based speech intention understanding and its application to in-car spoken dialogue system. In *Proceedings of the 19th International Conference on Computational linguistics* (Vol. 1), Taipei, Taiwan.

Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, Stanford, CA, USA.

Paek, T., & Pieraccini, R. (2008). Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communication*, *50*(8), 716–729.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI'03)*, Acapulco, Mexico.

Pineau, J., West, R., Atrash, A., Villemure, J., & Routhier, F. (2011). On the feasibility of using a standardized test for evaluating a speech-controlled smart wheelchair. *International Journal of Intelligent Control and Systems*, *16*(2), 124–131.

Png, S. & Pineau, J. (2011). Bayesian reinforcement learning for POMDP-based dialogue systems. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'11)*, Prague, Czech Republic.

Png, S., Pineau, J., & Chaib-Draa, B. (2012). Building adaptive dialogue systems via bayes-adaptive POMDPs. *IEEE Journal of Selected Topics in Signal Processing, 6*(8), 917–927.

Rabiner, L. R. (1990). Readings in speech recognition. In Chapter A tutorial on hidden Markov models and selected applications in speech recognition (pp. 267–296). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL'00)*, Hong Kong.

Thomson, B. (2009). Statistical methods for spoken dialogue management. *PhD thesis*, Department of Engineering, University of Cambridge.

Weilhammer, K., Williams, J. D., & Young, S. (2004). *The SACTI-2 corpus: Guide for research users*. Cambridge University. Technical report.

Williams, J. D. (2006). Partially observable Markov decision processes for spoken dialogue management. *PhD thesis*, Department of Engineering, University of Cambridge.

Williams, J. D., & Young, S. (2005). *The SACTI-1 corpus: Guide for research users*. Department of Engineering, University of Cambridge. Technical report.

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

Zhang, B., Cai, Q., Mao, J., Chang, E., & Guo, B. (2001a). Spoken dialogue management as planning and acting under uncertainty. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech'01)*, Aalborg, Denmark.

Zhang, B., Cai, Q., Mao, J., & Guo, B. (2001b). Planning and acting under uncertainty: A new model for spoken dialogue system. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI'01)*, Seattle, Washington, USA.