





An Adaptive and Transferable Dialog Management System for Social Aware Task Execution

Antonio Capela¹, Samuel Mascarenhas¹, Pedro Santos¹, and Manuel Lopes¹[0000-0002-6238-8974]

INESC-ID, Instituto Superior Técnico, Lisboa, Portugal

Abstract. Efficient and acceptable A.I. agents need to interact and dialog with their users taking into account not just task efficiency but also social preferences in the interaction. In this work we introduce an hierarchical model for generating dialog in an A.I. performing tasks such as delivering drinks in a bar or books in a library. At the base of the system is a dialog management based on POMDP (Partially Observable Markov Decision Process) models. This task specific component can by itself be used to execute the task. We then introduce another level, and define the interface between the levels, to be able to complement the dialog generated to take into account the social preferences of interaction of each particular user. We show how a particular parameterization of the state allows to learn a personalised policy. We further show that with our formalism a social policy learned for a particular user can then be used in other similar tasks without requiring further learning. We present several simulations showing how we can plan multiple tasks, learn social policies and that those policies can be transferred.

Keywords: Social Aware Agents · Dialog Management · Adaptive Learning · User Adaptation · Task Transfer

1 Introduction

A Spoken dialog system (SDS) is a computer system allowing people achieve a task by interacting with computers via spoken language to achieve their goals. SDSs are an increasingly prominent part of our society, we see them in costumer support, intelligent home appliances, automated personal assistants, and much more. The relatively recent increase in popularity of SDS makes them a very interesting topic to study.

With the current advances in AI the number of situations where we will be interacting with autonomous agents is expected to rise in the future. One example would be fully autonomous vehicles that provide taxi services. In this situation, how should an autonomous taxi interact with passengers from a social perspective? What happens when a person tries to abide by the same social norms he or she would when interacting with a human cab driver but the system is unable to reciprocate? In that case, the interaction will likely not be as pleasant

when compared to an autonomous driver that is capable of understanding and follow the same social conventions. These implicit conventions are described in sociological theories of interaction such as the highly influential Politeness theory [3]. This theory postulates that people use specific politeness strategies when communicating with each other in order to maintain each others social and moral standing, also referred to as the concept of 'face' in the theory. Moreover, rational agents, according to the theory, choose what they perceive to be the most effective strategies to avoid losing their face. This choice is non-trivial as, for instance, an overuse of the same strategy might come across as insincere and neglectful of the other person.

With the rise of smart speakers like Amazon Alexa, we are currently witnessing a renewed interest in spoken dialogue systems. The great progress made in speech recognition and speech synthesis opens up the potential for such systems to engage with their users in novel ways that go beyond simple information exchange. Moreover, the research fields of Intelligent Virtual Agents and Social Robotics are also highly invested in developing dialogue systems with social capabilities. The main reason is that there are several benefits in having a dialogue system that is capable of engaging in social dialogue with humans [1]. Not only this type of dialogue can make the interaction feel more enjoyable but it also can contribute to a greater feeling of trust in the system [2]. This is because human-human communication is greatly affected by how well the interlocutors are able to establish rapport with each other and studies have shown that the same applies to human-agent communication [6, 4]. In fact, in scenarios that involve the disclosure of sensitive information, such as a medical interview, people can feel even more comfortable when interacting with a virtual agent that is fully autonomous rather than one that is being operated by a human [5].

Initial strategies to model SDSs were based around creating a Finite State Automaton (FSA)[8, 9], which can be represented as a graph where the nodes represent prompts given by the machine and the links give the user's possible responses. In order to avoid errors in automatic speech recognition (ASR), the possible responses must be very limited, and the resulting dialogues could be very frustrating to the users. Different strategies appeared to deal with this but, as shown by Williams and Young[10], most fall under the POMDP framework.

In this paper we hope to achieve a machine capable of doing simple social tasks taking in consideration the social context and social preferences of each user. To do this the model will contain the usual POMDP model that is indifferent to the social implications of the conversation, only focused on the user's goals, with an additional layer is built on top of it, that is responsible for adding actions of social character, like greeting or thanking. This layer avoids using the internal structure of the POMDP in order to be possible to transfer it to a different problem where it would be able to perform without extra learning necessary.

This approach differs from existing work because of the separation between the task level planing, handled by a POMDP, and the social planing, handled by a layer on top of that.

Due to our contributed architectural choices and the defined interfaces between the different state representation, it is possible to use Reinforcement learning on the social layer to learn the social context and preferences of the user and transfer the social policy between different tasks.

2 Computational Model

The proposed decision architecture, shown in Fig.1, is divided into two main components. At the core, we have a model for the conversation that ignores social context and its only focus is on completing the task at hand. The POMDP model is used for this effect. This layer is obviously task specific and normally all its parameters can be hand-tuned based on the task.

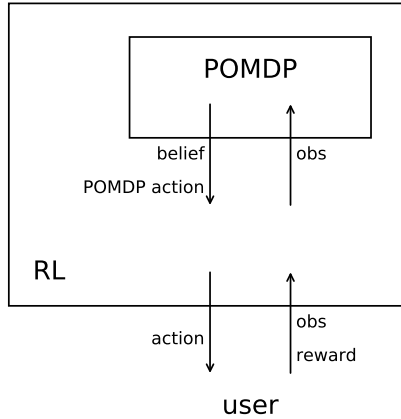


Fig. 1. Architecture of the model. At the core we have a POMDP responsible to generate a task specific dialog. At the top layer we have an RL agent that learns social policies that enhances the POMDP policy to take into account the social preferences of the users. An interface between the two levels was develop to ensure that the RL agent can learn and that its policy can be transferred to other tasks.

On top of that, we build a system that is responsible for adding the social interactions to accommodate the user. This system will use reinforcement learning to find the optimal policy, so we'll refer to it as the RL layer. This component is not task specific but is instead user specific. It will learn how each user prefers its interactions.

A great challenge is to provide clear interfaces between the task and social levels to ensure that: i) each component can work reliably; ii) the interface is as simple as possible; iii) the system can adapt to each user; iv) user dependent knowledge can be transferred.

With these goals in mind we defined the following interface. The RL layer will receive information about the change in belief of the POMDP because we want to know if the last part of the dialog changed the belief about the task. It also receives the optimal action proposed by the POMDP that the RL level can choose to accept or to overwrite it with a social action. The reason for this is that if the belief changed in the last interaction maybe the agent is in the middle of asking some task specific information and so we should not interrupt the exchange with social cues. For instance if the belief grew to increase the probability of the task being acquiring a beer, we should not interrupt to compliment the bar before asking if it is with or without alcohol. The RL layer will then observe the response from the user and, if it kept the action from the POMDP, it will relay it to the POMDP and update the belief.

We now go into more detail how these parts work.

2.1 Mathematical Models

POMDP - For Dialog Management A Partially Observable Markov Decision Process (POMDP) is a mathematical framework that can be used to model sequential decision problems where there is only partial information of the state of the environment. A POMDP is formally defined by the tuple $(S, A_m, T, R, O, Z, \lambda, b_0)$, where:

- S is the set of states
- A_m is the set of action the agent may take
- T defines a transition probability $p(s'|s, a_m)$
- R defines the expected reward $r(s, a_m)$
- O is the set of observations
- Z defines the observation probability $p(o|s', a_m)$
- λ is a geometric discount factor $0 \leq \lambda \leq 1$
- $b_0(s)$ is the initial state distribution.

At each time step t , the environment is in some state s . The agent chooses an action a_m that causes the environment to transition to state s' with probability $p(s'|s, a_m)$. Finally, the agent receives an observation o with probability $p(o|s', a_m)$ and a reward r equal to $r(s, a_m)$.

A useful quantity to keep track is the probability distribution over the states, known as the belief (b). This quantity can be updated according to equation Eq. (1), where $p(o'|a_m, b_t)$ is a normalising constant.

$$\begin{aligned}
 b_{t+1}(s') &= p(s'|o', a_m, b_t) \\
 &= \frac{p(o'|s', a_m, b_t)p(s'|a_m, b_t)}{p(o'|a_m, b_t)} \\
 &= \frac{p(o'|s', a) \sum_{s \in S} p(s'|a_m, b_t, s)p(s|a_m, b_t)}{p(o'|a_m, b_t)} \\
 &= \frac{p(o'|s', a) \sum_{s \in S} p(s'|s, a_m)b_t(s)}{p(o'|a_m, b_t)}
 \end{aligned} \tag{1}$$

The goal of the agent is to maximise the cumulative, infinite-horizon, discounted reward Eq. (2):

$$\Theta = \sum_{t=0}^{\infty} \lambda^t r_t \tag{2}$$

The optimal policy, that is, the policy that will maximise the expected return, will depend on the complete history of the dialog. However, the belief has a useful property that it is the complete summary of the dialog history. Formally, given an initial belief b_0 and a history $\{a_1, o_1, a_2, o_2, \dots\}$, the belief provides a sufficient statistic. As a result, the policy can be seen as simply a mapping from belief state to action $\pi(b) \in A_m$.

Spoken Dialog System-POMDP In this section we show how a POMDP is commonly used for generating Spoken Dialog Systems (SDS). We follow the approach of Williams and Young[10] The properties of spoken dialog make it so that, when represented as a POMDP, the state can be naturally separated into three distinct components:

- User goal (g)
- User action (u)
- Dialog history (h)

As a result, the factored POMDP state will be defined as:

$$s = (g, u, h) \tag{3}$$

With this in mind, we can expand the transition function and decompose it into its three components:

$$\begin{aligned} p(s'|s, a_m) &= p(g', u', h'|g, u, h, a_m) \\ &= p(g'|g, u, h, a_m)p(u'|g', g, u, h, a_m)p(h'|g', u', g, u, h, a_m) \end{aligned} \tag{4}$$

We can then simplify this equation by taking some independence assumptions. For the first term, which refers to how the user goal changes at each step, we'll assume it only depends on the previous goal, the dialog history and the machine's action.

$$p(g'|g, u, h, a_m) = p(g'|g, h, a_m) \tag{5}$$

The next term, how the user will act at each step, we assume to only depend on the current goal, the dialog history and the machine's action.

$$p(u'|g', g, u, h, a_m) = p(u'|g', h, a_m) \tag{6}$$

For the last term, which captures relevant information about the history of the dialog, this will only depend on the most recent variables, as well as the previous dialog history.

$$p(h'|g', u', g, u, h, a_m) = p(h'|g', u', h, a_m) \quad (7)$$

Replacing Eq. (5), Eq. (6) and Eq. (7) into Eq. (4), we get:

$$p(s'|s, a_m) = p(g'|g, h, a_m)p(g'|g', h, a_m)p(h'|g', u', h, a_m) \quad (8)$$

As for the observation, this will only depend on the user's action, so the observation probabilities become:

$$\begin{aligned} p(o'|s, a_m) &= p(o'|g, u, h, a_m) \\ &= p(o'|u) \end{aligned} \quad (9)$$

Combining Eq. (8) and Eq. (9) with Eq. (1), we get the equation for belief updating for the SDS-POMDP:

$$\begin{aligned} b(g', u', h') &= \\ &= k \cdot p(o'|u') \sum_{h \in H} p(u'|s', h, a_m) \cdot p(h'|g', u', h, a_m) \sum_{g \in G} p(g'|g, h, a_m) \sum_{u \in U} b(g, u, h) \end{aligned} \quad (10)$$

Overall, the SDS-POMDP framework makes it easier to model spoken dialog systems when compared to the usual POMDP framework, and the assumptions made allow more efficient algorithms.

RL - For User's Personality Adaptation The RL layer runs on top of the POMDP. Its task will be to choose between the action selected by the optimal policy (π_{POMDP}) of the POMDP or a social action, e.g. saying hello. It is composed of:

- The observed state s in which we will learn, composed of:
 - the current POMDP belief b .
 - the previous POMDP belief pb .
 - the action selected using the optimal policy from the POMDP a_m .
 - the previous action pa .
- A set of actions the agent can take, both the action from the POMDP and the social actions: $A = A_m \cup A_{social}$
- a hidden state h that is unknown and includes the POMDP state (of which the structure is known), as well as information on the user's social preferences. This could be information on whether the user is happy with the conversation so far, if he feels that he has been treated fairly, and so on.
- A reward function $r(h, a)$
- A observation probability function $p(o|h, a)$

As explained before this state guarantees that the agent can learn, but also that it can transfer the learned knowledge about the user for another task.

The update of the beliefs (b and pb) are dependent on the type of action a . If the action was social, both beliefs remain the same. If the action was not social, then b is updated using Eq. (10), and pb is set to b . Also, pa is set to the previous action, unless the last action terminated the dialog, in which case pa indicates that this is the beginning of a new dialog.

In order to have the RL layer not depend on the underlying POMDP structure, so that it can be transferred to another problem, instead of taking the belief directly, only the total variation (TV) between the current and the previous belief, calculated according to Eq. (11), will be used to create the optimal policy for the RL layer.

$$TV(b_{t+1}, b_t) = \frac{1}{2} \sum_{s \in S} |b_{t+1}(s) - b_t(s)| \quad (11)$$

An identifier (id) will also be used for as information on the last action. It will store the last social action that was performed (or whether it is the beginning of the conversation), and in the case of the last action being non-social, whether it is the same as the last non-social action or not. These two elements will be converted to a numerical vector ($\phi(s)$). The first component of the vector will be the total variation referred above and the rest will be a one-hot encoding of the id .

We can then create a policy using this encoding, by taking the inner product with a parameter vector \mathbf{w}_a for each social action plus one for keeping the optimal POMDP action, and then normalising using the softmax function, like in Eq. (12). The elements of the final vector give the probability of selecting the corresponding action.

$$\pi(s, a) = \frac{e^{-\mathbf{w}_a \cdot \phi(s)}}{\sum_{a' \in A} e^{-\mathbf{w}_{a'} \cdot \phi(s)}} \quad (12)$$

To find the parameters that optimise the policy, we use the evolution strategy CMA-ES [7]. The function to optimise is the sum of the rewards received over n -steps, and the optimisation occurs over the parameters \mathbf{w}_a .

3 Experiments

In the experiments we show the two main capabilities of our system. First that the system is able to learn how to use social actions to increase how well the user enjoys the interaction. We consider different user profiles and see how well it can adapt to each profile. Second we want to show that with our architecture we are able to transfer the knowledge about the user from one scenario to another. If an user enjoys a certain type of social interaction in one scenario we can use such preference to bootstrap the social actions in a new (related) scenario.

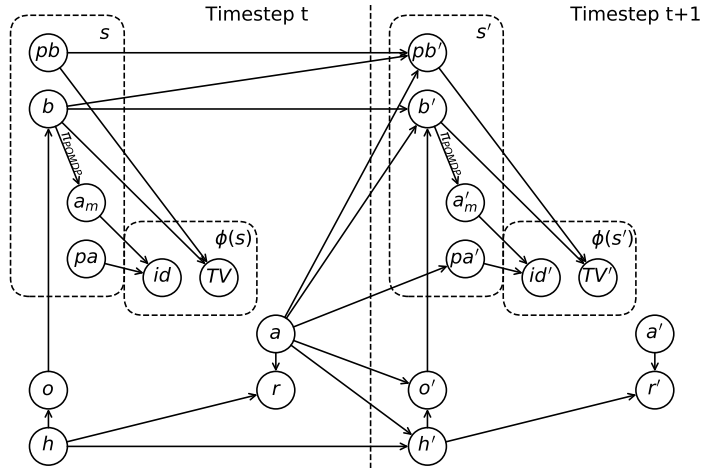


Fig. 2. Structure of the RL layer. Each arrow indicates that the variable is directly dependent on the other. The dashed boxes represent the observed state s and the numerical representation of the state $\phi(s)$. o is the observed response by the user, a is the action selected by the RL, r is the reward received. The other variables are explain in the text.

3.1 Scenarios

We consider two different scenarios to validate our claims and contributions. We need two scenarios to test the transfer and also to see how well the architecture we have is agnostic to the specific details of each task. In the following, we make a simple description of the POMDP structure of these scenarios:

Scenario 1: Bar In the first scenario, we have a customer going to a bar and asking for a drink. He can want a bottle of water, or a glass of beer of brand 1 or brand 2. The agent (acting as the barman) can ask the customer what he wants to drink, ask him to specify the brand of beer he wants or give the customer one of the three options, ending the dialog. At each time step, the agent has a probability p_{err} of misunderstanding the client and observing a different random action. For this POMDP, whenever the agent chooses to ask the customer what he wants or to specify he receives a small negative reward (-1). When giving one of the drinks, if the drink corresponds to the customer desired drink, then the agent receives a positive reward (10), otherwise, it receives a large negative reward (-100).

Scenario 2: Library In the second scenario, we have a user who goes to a library with the intention to request one of two books. The library however only has book 2, as a result, if the reader asks for book 1 then the agent (the librarian) must inform that there is no such book in the library, at which point the reader changes its mind and want book 2. Like in scenario 1, there is a probability

of misunderstanding p_{err} . As for the POMDP reward, the agent gets a small negative reward for asking what the reader wants (-1). If the agent gives book 2, but the reader still wants book 1, then it receives a large negative reward (-100). If the agent gives book 2 and the reader wants it, then it gets a positive reward (5) that is larger (10) if the reader wanted book 2 from the beginning .

Social components In any of the scenarios the agent can choose to include in the dialog not only the task relevant queries as described before, but also social components. For instance, greetings, e.g. hello/goodbye, or thanking. Including smalltalk is also important in some contexts but it is left for future work.

3.2 Clients personalities

To show how well we can adapt to different personalities we will create 2 different clients: a social (that likes interactions that use standard politeness formulas) and an anti-social (that just cares about the task efficiency).

We model this by the use of different reward functions for the RL layer. To have the reward be analogous something like tipping or a survey at the end, the reward will be zero if the action of the RL layer is not terminal, that is, it does not cause the conversation to restart. In both scenarios, a positive reward (1.) will be awarded if the agent perform the action that is expected of him, and a negative reward (-1.) if he fails (by giving the wrong drink or the wrong book). The difference between the two costumers is that the social costumer will receive an extra positive reward (.5) if at the beginning of the conversation the agent said hello, while the anti-social is indifferent to this.

Sample conversations for both scenarios and clients can be seen in section 5.

Check Learning Capability Four different policies, one for each combination of scenario and client personality, are learned using the method referred in 2.1. $\pi_{B,S}$ corresponds to the policy learned on the bar scenario with the social client, $\pi_{L,A}$ corresponds to the policy learned on the library scenario with the anti-social client, etc.

As can be seen in Fig. 3, the method used for optimisation is capable of improving the initial policy, created with random parameters. Due to the randomness of the observations (agent has a chance to misunderstanding what the user was saying), the received reward can be different from the expected reward. This helps explain the noise that can be seen in the objective function.

Check Transfer Capability To check the transferability of the policies, that is, if a social policy learned in one scenario will perform well in another, a series of simulations were performed on each scenario for all policies referred above. The results can be seen in table 1. As can be seen, policies trained with the same client profile perform very similarly, which means that even though a policy is trained in a different scenario, it achieves a performance close to the performance of the optimal policy trained in that scenario. We can conclude that for this example, the policy learned in one scenario transfers well to the other scenario.

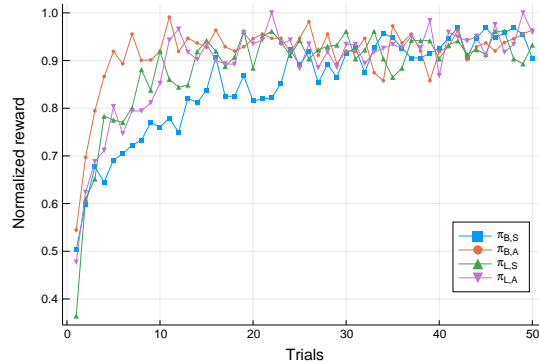


Fig. 3. Average reward of the μ best solutions with the number of trials during the training. The rewards are normalised so that the maximum of each one is 1.

It can also be observed that the policies learned from one client perform worse on the other client than the policies learned on that client. This is evidence that the optimal policy for both personalities is different.

Table 1. Estimated normalised average rewards and its standard deviation, for every policy in each scenario. In each row, the values are normalised such that the reward for the social policy trained in the respective Scenario/Client is 1. We can clearly see that our mechanism for transfer works as social policies learned in one task can be executed in a different task and achieve a high success if the social profile of the user is the same.

Scenario/Client	$\pi_{B,S}$	$\pi_{B,A}$	$\pi_{L,S}$	$\pi_{L,A}$
Bar/Social	1.0 \pm 0.0	0.784 \pm 0.003	0.995 \pm 0.004	0.787 \pm 0.003
Bar/Anti-social	0.767 \pm 0.003	1.0 \pm 0.0	0.771 \pm 0.004	1.014 \pm 0.005
Library/Social	1.005 \pm 0.004	0.810 \pm 0.003	1.0 \pm 0.0	0.810 \pm 0.003
Library/Anti-social	0.758 \pm 0.003	0.997 \pm 0.004	0.756 \pm 0.003	1.0 \pm 0.0

4 Conclusions

We have defined a model that separates the social and task planning into two components. The task planning is formally modelled as a POMDP that has been shown to be a framework appropriate for planning in SDSs. In particular, a variation on the POMDP referred to as SDS-POMDP is used, that takes advantage of the properties of SDS to achieve more efficient results. The social planning comes from a layer on top of the POMDP that will receive information about the change in belief of the POMDP and the optimal action proposed by the POMDP. Since the belief is not used directly, and instead only the observed

change in belief is used, the policies created are independent of the underlying POMDP structure. RL is then used to learn a policy that selects whether to override the POMDP action with a social action or to accept such action.

This approach allows for the social policy to not depend on the underlying structure of the POMDP, and as a result, the social policy learned in a particular scenario can be transferred to another.

We trained the RL layer on two different simulated scenarios with two different user personalities, and found that it was capable of learning the policies that optimise the rewards. We then simulated the policies on scenarios different from the one they were trained in and found that the learned policies still performed well.

In this work, social interactions were fairly minimal. More complex interactions like smalltalk could be added and would be interesting to see to what extent this would be transferable between tasks. It would also be of interest to see how the model would perform on real users and not just simulations, but for that the necessary empirical data would need to be gathered.

5 Appendix

Sample conversations for both scenarios from section 5. In parentheses are the observed user actions. Due to the change of misunderstanding the client (dictated by p_{err}), the system repeats some questions to assure that it understands the users intentions.

Bar scenario with anti-social client. The system follows policy $\pi_{B,A}$.	Library scenario with social client. The system follows policy $\pi_{L,S}$.
s: What would you like?	s: Hello!
u: I'd like a beer please.(beer)	u: Hello.(hello)
s: We have brand1 and brand2. Which would you prefer?	s: What would you like?
u: Brand1 please.(Brand1)	u: I would like to order book1(book1)
s: We have brand1 and brand2. Which would you prefer?	s: What would you like?
u: Brand1 please.(Brand1)	u: I would like to order book1(book1)
s: Here is your beer. Enjoy.	s: Sorry, we don't have that book.
	u: I would like to order book2(book2)
	s: Here is book2

Acknowledgements

This work was partially by the FCT grants FCT: UID/CEC/50021/2019 and PTDC/CCI-COM/30787/2017.

References

1. André, E., Pelachaud, C.: Interacting with embodied conversational agents. In: Speech technology, pp. 123–149. Springer (2010)

2. Bickmore, T., Cassell, J.: Relational agents: a model and implementation of building user trust. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 396–403. ACM (2001)
3. Brown, P., Levinson, S.C.: Politeness: Some universals in language usage, vol. 4. Cambridge university press (1987)
4. Cassell, J., Gill, A.J., Tepper, P.A.: Coordination in conversation and rapport. In: Proceedings of the workshop on Embodied Language Processing. pp. 41–50. Association for Computational Linguistics (2007)
5. Gratch, J., Lucas, G.M., King, A.A., Morency, L.P.: It’s only a computer: The impact of human-agent interaction in clinical interviews. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems. pp. 85–92. AAMAS ’14, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2014), <http://dl.acm.org/citation.cfm?id=2615731.2615748>
6. Gratch, J., Wang, N., Gerten, J., Fast, E., Duffy, R.: Creating rapport with virtual agents. In: International workshop on intelligent virtual agents. pp. 125–138. Springer (2007)
7. Hansen, N., Ostermeier, A.: Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The (=). *Eufit* **97**, 650–654 (1997)
8. McTear, M.F.: Modelling spoken dialogues with state transition diagrams: experiences with the cslu toolkit. In: Fifth International Conference on Spoken Language Processing (1998)
9. Sutton, S., Novick, D.G., Cole, R., Vermeulen, P., de Villiers, J., Schalkwyk, J., Fenty, M.: Building 10,000 spoken dialogue systems. In: Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP’96. vol. 2, pp. 709–712. IEEE (1996)
10. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language* **21**(2), 393–422 (2007)