

MODELLING USER BEHAVIOUR IN THE HIS-POMDP DIALOGUE MANAGER

S. Keizer, M. Gašić, F. Mairesse, B. Thomson, K. Yu, and S. Young

Department of Engineering, University of Cambridge (UK)

ABSTRACT

In the design of spoken dialogue systems that are robust to speech recognition and interpretation errors, modelling uncertainty is crucial. Recently, Partially Observable Markov Decision Processes (POMDPs) have shown to provide a well-founded probabilistic framework for developing such systems. This paper reports on the design and evaluation of the user act model (UAM) as part of the Hidden Information State (HIS) POMDP dialogue manager. Within this system, the UAM represents the probability of a user producing a certain dialogue act, given the last system act and the dialogue state. Its design is founded on the notions of adjacency pairs and dialogue act preconditions and it is domain-independent. Experimental evaluation results on both simulated and real data show that the UAM plays a significant role in improving robustness, but it requires that the N-best lists of user act hypotheses and their confidence scores are of good quality.

Index Terms— Spoken dialogue systems, POMDPs, user modelling, evaluation

1. INTRODUCTION

One of the major challenges in the development of spoken dialogue systems is the problem of uncertainty due to speech recognition and language understanding errors, and unexpected user behaviour. Most systems rely on choosing the most likely result from ASR and NLU, and in cases where the chosen input turns out to be incorrect, complex repair strategies are required. By modelling dialogue as a POMDP (Partially Observable Markov Decision Process), a framework is provided that enables these uncertainties to be modelled in a well-founded (probabilistic) manner thereby leading to more robust policies [1].

A key component of the POMDP framework is the *user act model* (UAM) $P(a_u | s_m, a_m)$, which represents the probability that the user would generate a user act a_u , given the last system act a_m and the system state s_m . In this paper, we focus on the design of the UAM in the Hidden Information State (HIS) POMDP system and provide experimental evaluations of its effectiveness.

In Section 2, the HIS POMDP system is briefly reviewed, followed in Section 3 by a more detailed discussion of the UAM. In Section 4 evaluation results on both simulated and real data are presented, showing the effectiveness of the UAM. Finally, Section 5 gives some conclusions.

This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSiC project: www.classic-project.org).

2. THE HIS-POMDP DIALOGUE MANAGER

The architecture of a POMDP-based dialogue system is shown in Fig. 1. The user produces an action a_u based on his goal s_u , resulting in a speech signal to be analysed by the system’s speech understanding component. This results in an N-best list of user action hypotheses a_u^1, \dots, a_u^N . Instead of just taking the 1-best result to update the dialogue state s_m , the entire N-best list with confidence scores is used to compute a probability distribution over all possible dialogue states, the belief state $b(s_m)$.

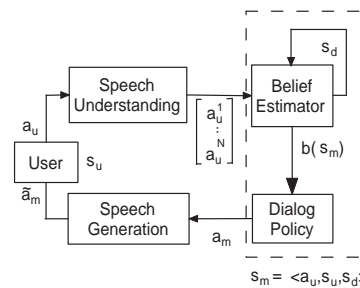


Fig. 1. Architecture of a POMDP-based spoken dialogue system

Based on the current belief state b , the machine selects an action a_m , receives a reward $r(s_m, a_m)$, and transitions to a new (unobserved) state s'_m , where s'_m depends only on s_m and a_m . The machine then receives an observation o' consisting of a new N-best list of hypothesised user actions. Finally, the belief distribution b is updated based on o' and a_m .

Although it provides a powerful and well-founded way of modelling uncertainty, the POMDP approach to dialogue management introduces problems of computational complexity, both for belief monitoring and policy optimisation. The Hidden Information State (HIS) POMDP system deals with these problems in several ways [2].

First, the dialogue state is factored into three components: the last user act a_u , the user’s goal s_u and the dialogue history s_d . The dialogue history is based on a simple grounding model and is encoded deterministically. It yields probability one when the updated dialogue state hypothesis is consistent with the history, and zero otherwise.

Second, the space of user goals is partitioned into equivalence classes p of equally likely goals. By replacing s_m by its factors (s_u, a_u, s_d) and making reasonable independence assumptions, it can be shown that in partitioned form, the belief state is updated as follows:

$$\begin{aligned}
b'(p', a'_u, s'_d) = k \cdot & \underbrace{P(o'|a'_u)}_{\substack{\text{observation} \\ \text{model}}} \cdot \underbrace{P(a'_u|p', a_m)}_{\substack{\text{user action} \\ \text{model}}} \\
& \cdot \sum_{s_d} \underbrace{P(s'_d|p', a'_u, s_d, a_m)}_{\substack{\text{dialogue model}}} \cdot \underbrace{P(p'|p) \cdot b(p, s_d)}_{\substack{\text{belief refinement}}}
\end{aligned} \quad (1)$$

where p is the parent of p' , i.e., p' is a partition resulting from splitting p .

In this equation, the *observation model* is approximated by the normalised distribution of confidence measures output by the speech understanding system. Unlike the observation model, the *user action model* takes the context in which the user performed his action into account. It allows the observation probability that is conditioned on a'_u to be scaled by the probability that the user would perform the action a'_u given the partition p' and the last system prompt a_m . For example, the observation model might not be able to distinguish between an acknowledgement and an affirmation for the utterance “Yes”, but if the last system act was a confirmation (“You want Chinese food?”), the affirmation reading should become more likely.

3. THE HIS USER ACT MODEL

The HIS user action model is a hybrid model, consisting of a dialogue act type *bigram model* and an *item matching model*:

$$P(a'_u|p', a_m) \approx \underbrace{P(\mathcal{T}(a'_u)|\mathcal{T}(a_m))}_{\substack{\text{bigram model}}} \cdot \underbrace{P(\mathcal{M}(a'_u)|p', a_m)}_{\substack{\text{item matching model}}} \quad (2)$$

where $\mathcal{T}(\cdot)$ denotes the *type* of the dialogue act and $\mathcal{M}(\cdot)$ denotes whether or not the dialogue act *matches* the current partition p' and the last system act.

User goals, and therefore, partitions, are specified in terms of slot-value pairs, e.g., (`food=Chinese`). A domain ontology specifies a hierarchical structure among the slots, allowing slots to be defined for specific entities only (e.g., restaurants have a food type, but bars do not). Dialogue acts take the form `actt(a=v)` where *actt* is the dialogue act type, *a* is an attribute or slot and *v* is its value (for example, `inform(food=Chinese)`).

3.1. The bigram model

The bigram model reflects the dialogue phenomenon of *adjacency pairs* [3]. For example, a question is typically followed by an answer, an apology by an apology-downplayer, and a confirmation (“you want Chinese food?”) is typically followed by an affirmation (“Yes please.”), or negation (“No, I want Indian.”).

The bigram model is trained from data using maximum likelihood. A corpus of dialogues obtained from a recent trial involving several statistical dialogue systems including the HIS system [4] was used to estimate the probabilities. Since many combinations of dialogue act types are never observed, Witten–Bell smoothing was applied to deal with data sparsity.

3.2. The item matching model

The item matching model reflects the probability that a user act is consistent with the partition and the last system act. Each partition represents a hypothesised, possibly underspecified user goal. For example, `find(restaurant(food=Chinese))` represents the set of all goals relating to a Chinese restaurant.

The item matching model is deterministic, assigning either a *full match probability* or a *no match probability*, depending on the outcome of matching the user act with the current partition. These matching probabilities are optimised empirically. For example, the user is not likely to ask about Indian food when the user goal actually indicates he wants a Chinese restaurant. Therefore, the act content (items, consisting of slot-value-pairs) of an inform should match the partition. On the other hand, a negation is not very likely if the content of the last system act matches the partition.

The design of the matching model is formalised in terms of dialogue act *preconditions* [5]. The preconditions of an action specify the conditions that the current dialogue state has to satisfy in order for an agent to perform that action. The user wanting to find a Chinese restaurant motivates him to perform the action `inform(type=restaurant, food=Chinese)` (assuming cooperativity in the sense that system will try to satisfy the user’s goal once it has been informed about it). Each precondition is defined in terms of an agent, a propositional attitude (typically ‘wants’), and a propositional argument (typically a slot-value pair). For example, `inform(food=Chinese)` has the precondition ‘U WANTS (food=Chinese)’, whereas a `negate()` after `confirm(food=Indian)` has the precondition ‘not WANTS (food=Indian)’.

In Table 1, some examples are given of dialogue acts and their preconditions in terms of propositional attitudes of the user, and what matching operations against a partition are required for these preconditions to be satisfied. Since the preconditions do not depend on the slots and their values that are specific to the domain, the item match model specification is domain-independent.

In some cases, the preconditions should include other kinds of properties than items from user and system acts to be matched against the partition. For example, the user is only likely to ask for alternatives if the system has offered a venue. This is the case for partitions in which the ‘name’ slot is instantiated. A similar argument can be made about `request(.)`: the user is only likely to ask for more information about a venue if such a venue is under discussion.

4. EVALUATING THE USER ACT MODEL

The user act model is evaluated in two ways. The first approach compares the overall system performance of the dialogue manager with the user act model included in Eq. 1 and with it excluded. The second approach investigates the user act model as an information source which can be used to rescore the N-best list of user acts as obtained from the speech understanding system.

User act	Preconditions	Matching
inform(a=x, b=y)	U WANTS a=x, b=y	a=x, b=y
request(a, b=x)	U WANTS b=x U WANTS U KNOWS_VAL a	b=x a
reqalts(a=x)	U WANTS a=x	a=x
confirm(a=x)	U WANTS U KNOWS_IF a=x	a=x
affirm()	[sys: confirm(a=x)] U WANTS a=x	a=x
affirm(b=y)	[sys: confirm(a=x)] U WANTS a=x U WANTS b=y	a=x b=y
negate()	[sys: confirm(a=x)] not (U WANTS a=x)	not (a=x)
negate(b=y)	[sys: confirm(a=x)] U BEL S BEL U WANTS a=x not (U WANTS a=x) U WANTS b=y	not (a=x) b=y

Table 1. Selection of user acts, their preconditions, and required matching operations; for cases in which it is relevant, the last system act [sys: act] is given. Further notation: KNOWS_VAL: ‘knows the value of’; BEL: ‘believes’; ‘**not**(a=x)’: the item (a=x) may not match the partition.)

4.1. System performance on simulated data

For the system evaluation approach, we used a user simulator [6] to run a large number of simulated dialogues with the dialogue manager and then measured the system performance in terms of average dialogue score and success rate. A dialogue is considered to be successful, if the correct venue has been offered and the additional information has been given, as specified in the predefined user goal. The score of a dialogue is obtained by subtracting 1 point for each turn and adding 20 points in case of a successful dialogue.

In Fig. 2, results are given comparing the success rates for the system with and without the UAM in operation and also with only the bigram component disabled. The results were obtained by running 3000 simulated dialogues with the system at different error rates for both configurations.

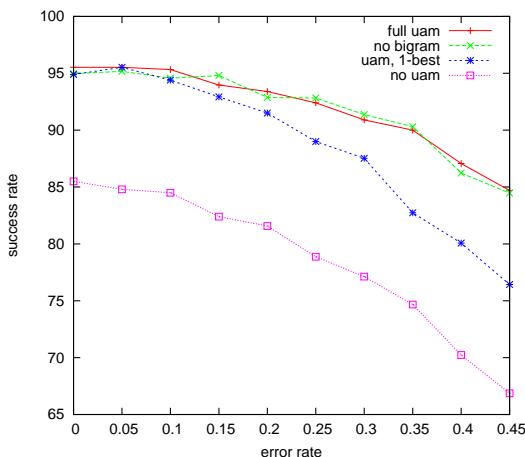


Fig. 2. Success rates for the HIS DM, comparing different configurations of the UAM.

The results clearly show a dramatic improvement in success rate when using the UAM (the dialogue scores not given here for lack of space show the same tendency). However, the bigram model does not to have a significant impact on performance, since the scores are very similar when the bigram distributions are set to uniform. Fig. 2 also compares the success rates of using 1-best and 2-best lists of simulated user acts. These results show that the system can benefit more from the UAM by using multiple user act hypotheses instead of just the 1-best, especially at the higher error rates.

4.2. Evaluating rescoring of N-best lists on real data

For the semantic evaluation approach, the effect the UAM has on the N-best list of user acts and their probabilities generated by the speech understanding system was investigated. Since the UAM determines how likely user acts are given the current dialogue state, and the semantic decoder does not take the dialogue state into account, N-best lists that are rescaled by the UAM should be more accurate.

Using the trial data mentioned in Section 3, we have evaluated the semantic decoder output before and after applying the user act model. The trial systems all used the same speech understanding component, in which the speech recogniser generated 10-best lists of utterances, which were then semantically decoded into N-best lists of typically no more than 3 user act hypotheses. For each user act a'_u in the N-best list, the original observation probability $P(a'_u|o')$ was rescored by multiplying it with the user act model probabilities $P(a'_u|p'(h), a_m)$, weighted by the current beliefs of the hypotheses h containing this user act:

$$P(a'_u|o') \cdot \sum_h P(a'_u|p'(h), a_m) \cdot P(h) \quad (3)$$

where $p'(h)$ denotes the partition associated with a particular hypothesis.

The semantic output is evaluated using different metrics that operate on the *semantic items* a dialogue act consists of, a semantic item being either a slot-value pair or the dialogue act type. The scores obtained are based on the number of item-level substitutions, insertions and deletions. The metrics considered are the oracle accuracy (OAcc), the accuracy of the top-ranked user act hypothesis (TAcc), the confidence-weighted recall (RCL) and precision (PRC), and the item-level cross entropy (ICE). The ICE metric was proposed in [7] as an effective measure for evaluating the usefulness of an N-best list of semantic hypotheses within a dialogue system (note that smaller ICE values indicate higher quality).

Table 2 gives the results both for using only the trial dialogues that were obtained with the HIS system (the top two rows in the table), and for all of the trial dialogues, regardless of the particular dialogue manager used (the bottom two rows).

For all of the metrics used, the rescored probabilities show an improvement over the original observation probabilities that is consistent over all datasets. However, in contrast to what might have been expected from the evaluations with simulated dialogues, the differences are marginal. The gain in performance that could be made by employing larger N-best

Resc	Corp	Turns	Acts	OAcc	TAcc	RCL	PRC	ICE
UAM	HIS	779	1.4	78.9	74.9	75.6	81.3	1.744
	HIS	779	1.4	78.9	75.3	75.8	81.5	1.739
UAM	all	4231	1.4	77.2	72.9	73.8	80.2	2.036
	all	4231	1.4	77.2	73.2	73.9	80.4	2.032

Table 2. Semantic evaluation results for trial data. Resc indicates whether or not the N-best list was rescored, Corp denotes the corpus of dialogues used, Turns denotes the number of user turns, Acts denotes the average number of act hypotheses in each N-best list per turn.

lists of user acts generated by the simulator is not reflected by the semantic evaluations on real data.

In order to relate the system performance results on the simulated data with the semantic evaluation results, we also ran 1000 simulated dialogues on different error rates and performed semantic evaluations on the N-best lists of simulated user acts and their rescored versions. The results in Table 3 show that the UAM now has a stronger impact, particularly on the ICE scores.

Resc	ER	Turns	Acts	OAcc	TAcc	RCL	PRC	ICE
NONE	15	8374	1.1	93.7	87.7	92.0	88.5	0.756
UAM	15	8374	1.1	93.7	88.3	92.1	98.0	0.138
UAM	30	9832	1.6	91.5	76.6	80.8	78.8	1.405
	30	9832	1.6	91.5	79.1	81.6	86.4	0.971
UAM	45	10231	1.7	90.3	73.1	77.4	75.7	1.689
	45	10231	1.7	90.3	76.0	78.4	83.0	1.275

Table 3. Semantic evaluation results for simulated dialogues. ER denotes the error rate of the simulated user acts.

From these evaluation results, one can conclude that the UAM can potentially make a significant difference in both system performance and obtaining more accurate N-best lists, but the nature of the N-best lists generated by the semantic decoder in the trial data is such that the UAM cannot do much to improve it. The acts in the simulated N-best lists [8] show more variation and have higher oracle scores, enabling the UAM to rescore them more effectively.

5. CONCLUSION

This paper has reported on the user act model within the HIS-POMDP dialogue manager. The model is a theoretically well-founded hybrid model, consisting of a bigram and an item matching model. The bigram model reflects the notion of adjacency pairs of dialogue act types, and can be trained from data, using maximum likelihood. In the design of the item matching model, the notion of dialogue act preconditions is used, thereby providing a theoretically well-founded and domain-independent procedure for determining the logical consistency of a user act w.r.t. the current dialogue state.

Experimental evaluations on simulated dialogues showed that the UAM plays a key role in improving robustness at higher error rates, provided good quality N-best lists are available. Since the gain in system performance on the simulated data can almost entirely be ascribed to the item matching model, the bigram model clearly needs improvement and we

are currently investigating longer span models in which previous user acts are included.

Evaluation results on real dialogues in terms of rescoring dialogue act hypotheses indicated that on N-best lists that have less variability and a relatively poor oracle accuracy, the UAM has only a small effect. This confirms our intuition that the generation of high quality N-best lists by the speech understanding system is crucial to overall dialogue system performance.

6. REFERENCES

- [1] J. Williams and S. Young, “Partially Observable Markov Decision Processes for spoken dialog systems,” *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.
- [2] S. Young, J. Schatzmann, K. Weilhammer, and H. Ye, “The hidden information state approach to dialog management,” in *Proc. ICASSP 2007*, Honolulu, Hawaii, 2007.
- [3] E. Schegloff and H. Sacks, “Opening up closings,” *Semiotics*, vol. 7, no. 4, pp. 289–327, 1973.
- [4] M. Gašić, S. Keizer, B. Thomson, F. Mairesse, J. Schatzmann, K. Yu, and S. Young, “Training and evaluation of the HIS-POMDP dialogue system in noise,” in *Proc. 9th SIGdial*, Columbus, Ohio, 2008.
- [5] P.R. Cohen and C.R. Perrault, “Elements of a plan-based theory of speech acts,” *Cognitive Science*, vol. 3, pp. 177–212, 1979.
- [6] J. Schatzmann, B. Thomson, K. Weilhammer, and H. Ye S. Young, “Agenda-based user simulation for bootstrapping a POMDP dialogue system,” in *Proc. HLT/NAACL*, Rochester, NY, 2007.
- [7] B. Thomson, K. Yu, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, and S. Young, “Evaluating semantic-level confidence scores with multiple hypotheses,” in *Proc. Interspeech*, Brisbane, Australia, 2008, To appear.
- [8] J. Schatzmann, B. Thomson, and S. Young, “Error simulation for training statistical dialogue systems,” in *Proc. ASRU*, Kyoto, Japan, 2007.