BAYESIAN DIALOGUE SYSTEM FOR THE LET'S GO SPOKEN DIALOGUE CHALLENGE

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

Cambridge University Engineering Department

ABSTRACT

This paper describes how Bayesian updates of dialogue state can be used to build a bus information spoken dialogue system. The resulting system was deployed as part of the 2010 Spoken Dialogue Challenge. The purpose of this paper is to describe the system, and provide both simulated and human evaluations of its performance. In control tests by human users, the success rate of the system was 24.5% higher than the baseline Lets Go! system.

Index Terms— POMDP, dialogue management, Let's Go

1. INTRODUCTION

The partially observable Markov decision process (POMDP) provides a statistical model for building spoken dialogue managers. Past researchers have used this model to build complete systems for tourist information [1], router troubleshooting [2] and voice dialling [3]. This paper describes a POMDP-based system built for the 2010 Let's Go spoken dialogue challenge, where the task is to provide bus information. This system is called BUDSLETSGO.

The key attributes of the system are:

- A universal statistical language model is used for speech recognition.
- The dialogue manager maintains a full probability distribution over all possible user goals at all times.
- At any stage, the user may provide any information. In this sense, the dialogue is mixed-initiative.

The paper is structured as follows. The POMDP dialogue model and its application to the Let's Go domain is described in section 2. The core components of the system are then described in the order that they are used in a turn of the dialogue. Section 3 describes the spoken language understanding component of the system, which converts the speech waveform into a collection of hypothesised semantic representations. These semantic representations are called *dialogue acts*. Next, the dialogue manager is described, which decides what the system should say in its turn. This is passed to the output generator, described in section 5, which converts the system output, first to text and then to a speech waveform. Section 6 describes an initial simulated evaluation of the system and section 7 concludes the paper.

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). Email addresses: {brmt2, fj228,mg436,sk561,farm2,ky219,sjy}@eng.cam.ac.uk

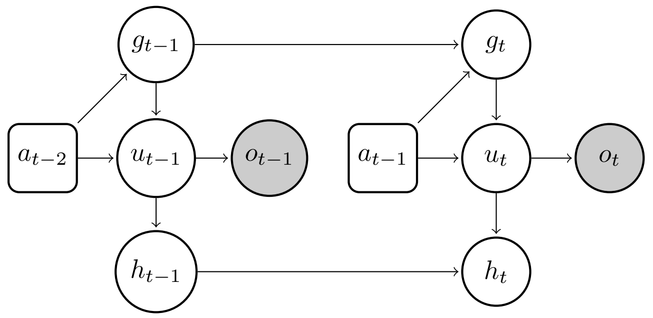

Fig. 1. Dependencies in the BUDS dialogue system.

2. THE DIALOGUE MODEL

The POMDP model of dialogue assumes that a dialogue consists of a sequence of turns, t. In each turn, the dialogue state, s_t, is hidden and depends on the previous state, s_{t-1}, and the system's previous action, a_{t-1}. The system's beliefs about the dialogue state, b_t, is a probability distribution over s_t. These beliefs are updated based on observations the system receives about the user, o_t, where o_t depends on s_t and a_{t-1}. In this paper, the observation is represented as an N-best list of user dialogue acts with associated confidence scores.

This paper makes use of a special case of the POMDP model, called the Bayesian update of dialogue state (BUDS) model [4]. In the BUDS model, the dialogue state is factorized into a user goal, g_t, the user's true dialogue act, u_t, and a history of the dialogue h_t. The user goal and dialogue history are further factorized into a series of *concepts*, c, with concept-level goals (called *sub-goals*) labelled $g_{t,c}$ and concept-level histories denoted $h_{t,c}$. Conditional independence assumptions are taken such that the $g_{t,c}$ depend on a_{t-1} and $g_{t-1,c}$ and optionally a parent goal, $g_{t,pa(c)}$. u_t depends on a_{t-1} and the $g_{t,c}$ and $h_{t,c}$ depend on $h_{t-1,c}$ and u_t. The dependencies between the unfactorized variables are shown in Figure 1.

2.1. The BUDSLETSGO system

The task for the BUDSLETSGO system is to provide information on bus timetables in Pittsburgh, PA. In each dialogue, the user will need to provide a place of departure, a destination and a time. To model this, the BUDSLETSGO system makes use of 16 concepts. A list of the concepts and their values is given in table 1.

Four of the concepts are used to model where the user is leaving from. The `from` concept models the type of place of departure, which can be a monument, neighborhood or stop. The other three concepts describe the exact location within the given type, and take the value "N/A" if the type is not applicable. These concepts are `fstop`, for when the departure point is a pair of road names, `fmon`, when the departure point is a monument, and `fneigh`, when the departure point is a neighborhood. Any pair of road names is allowed for `fstop`, including pairs which do not exist. The user is also allowed to specify that they would like to leave from "ANYWHERE" on a given road. The four concepts used for modelling the place of arrival are similar.

Six concepts are used for describing the user's required time of travel. The first of these is the `time` concept which is either "NEXT", for the next bus, or "time_specific", for a specific requested time. The remaining five concepts only apply when the time value is "time_specific". These are the `day`, `hour`, minute (`min`) and period of day (`pd`) of travel as well as the time reference (`tref`), which denotes whether the user wants to arrive before or after or leave before or after the given time.

The final two concepts used in the system are `meth` and `disc`. The `meth` node describes whether the user is asking for a bus with some constraints, is finished or wants to restart. `disc` models how the user issues "discourse" actions, which relate to only one turn in a dialogue. Example discourse actions are when the user asks for the following or previous bus. Figure 2 gives a Bayesian network of the dependencies between all the concepts. Each variable is applicable only if its parent's values are appropriately set.

Concept	Size	Example Values
`from`	3	ftstop, ftmonument, ftneigh
`fstop`	328 455	"FORBES&MURRAY", "ANYWHERE&FORBES"
`fmon`	52	"AIRPORT", "CENTURY SQUARE"
`fneigh`	220	"DOWNTOWN", "SQUIRREL HILL"
`to`	3	ttstop, ttmonument, ttneigh
`tstop`	328 455	"FORBES&MURRAY", "ANYWHERE&FORBES"
`tmon`	52	"AIRPORT", "CENTURY SQUARE"
`tneigh`	220	"DOWNTOWN", "SQUIRREL HILL"
`time`	2	"NEXT", time_specific
`hour`	12	"ONE", "TWELVE"
`min`	60	"ZERO", "TEN", "THIRTY FIVE"
`pd`	2	"AM", "PM"
`day`	10	"TODAY", "WEDNEDAY", "DAY AFTER TOMORROW"
`tref`	4	"ARRIVE BEFORE", "LEAVE AFTER"
`meth`	4	"RESTART", "FINISHED", constraints,
`disc`	9	"REPEAT", "FOLLOWING", "PREVIOUS", none

Table 1. Table of concepts in the extended BUDSLETSGO system. Values in lower case represent internal values, which the user cannot directly inform. The size column presents the number of applicable values for the concept. All concepts also include a "N/A" value for when the concept is not applicable.

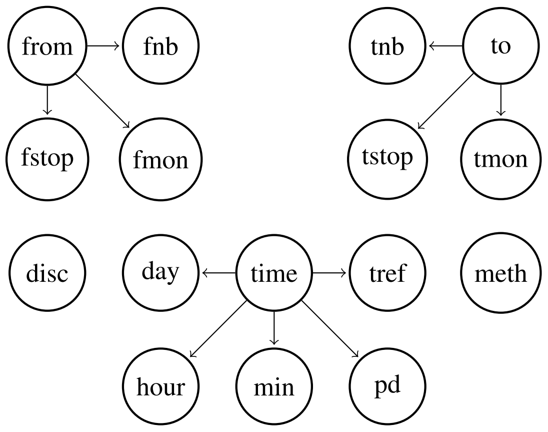

Fig. 2. Sub-goals in the BUDSLETSGO system.

3. SPOKEN LANGUAGE UNDERSTANDING

The BUDSLETSGO speech recogniser uses the ATK/HTK speech recognition toolkit [5]. As part of the 2010 spoken dialogue challenge, a corpus of live dialogues with the Olympus Let's Go system was released for all dialogues in the years 2003, 2005, 2008 and 2009 [6]. The BUDSLETSGO system uses a tri-gram statistical language model trained on this corpus along with pseudo-data generated from all possible concept values and simple Let's Go sentence patterns. The total number of words in the language modelling training corpus was 6.44M words. The size of the dictionary is 1700 words.

The acoustic model is a narrow-band model trained on

127.5 hours of data, of which around 40 hours of data are from the Wall Street Journal and tourist information domains and the remaining 87.5 hours of data are from the Let's Go corpus. The front-end uses perceptual linear predictor (PLP) features with energy and its first, second and third derivatives. A heteroscedastic linear discriminant analysis (HLDA) transform is used to project these features down to 39 dimensions. When tested on a held-out data set within the Let's Go domain (constructed from the 2005 data), the word error rate of the top hypothesis was 27.96%. In live operation, the system produces a 10-best of list of recognition hypotheses.

After converting the audio signal to words, the system must analyse the semantics of each utterance. This is implemented using a hand-crafted Phoenix parser, based on the one built for the Olympus Let's Go system [6]. Various minor modifications of the original system were made to convert the output of this parser into the Cambridge dialogue act format of [1]. For example, the original parser produced different semantic forms for the user saying "Going to Forbes" and "Going to Forbes Avenue". After transformation, both of these were represented as "inform(tstop=FORBES)". The Cambridge dialogue format structures the semantic representation as an act *type* followed by a sequence of *items*, where each item contains a concept and a value (both of which are optional). The concepts and their possible values are given in table 1, but are restricted to the values given in capital letters in the table.

4. THE DIALOGUE MANAGER

The dialogue manager bases all its decisions on the belief distribution over the possible user goals described in section 2. The mechanism used for updating is described in section 4.1, section 4.2 describes the belief state summary used for policy decision and a hand-crafted policy is described in section 4.3.

4.1. Belief updating

Belief updating is implemented using a grouped form of belief propagation [4]. Within each goal node, one enumerates only the values which have appeared in the output of the semantic decoder. All remaining values are grouped together and updated simultaneously. Full details of the general purpose updating algorithm are given elsewhere [4, 7].

The one special case that occurs in the BUDSLETSGO system is the updating of the `fstop` and `tstop` nodes. These two nodes represent stops as pairs of street names, where any pair of names is possible. The groups for the values for this node will be more complex than a simple list of values that have been observed in the user acts. Instead, the values are partitioned into groups including the given road name, and groups not including the road name. This partitioning is essentially the same as the partitioning of the hidden information state model [1], but is done at a node-level instead of for the entire dialogue state. As an example, when the user says that they would like to go to "Forbes Avenue and Murray", the partitions for the `tstop` goal will be "FORBES&MURRAY", "FORBES&OTHER", "MURRAY&OTHER", and "OTHER". The probability updates for these goals are the same as with the general case of grouped belief propagation discussed in previous work [4].

4.2. Summary state

All policy decisions are based on a summary of the full belief state. This summary is split into three parts: concept-level information, discourse information and informing information.

The concept-level information is formed as a sequence of summary vectors for each concept, c. For a particular concept, the system forms a tuple of the probability of the most likely value for that concept, p_{c1}, and the probability of the second most likely value for the concept, p_{c2}. This tuple is then mapped into a region number from 0 to 7, where the regions are the index of the closest point to (p_1, p_2) in the set $\{(0.9, 0)$, $(0.7, 0.2)$, $(0.7, 0.0)$, $(0.5, 0.4)$,, $(0.5, 0.2)$, $(0.5, 0.1)$, $(0.35, 0.35)$, $(0.1, 0.0)\}$. This is then joined with a binary feature to denote whether the probability that the concept is applicable given its parents is greater than 0.5.

The discourse information stores the probability of four goal values which are particularly important for handling the discourse. These are the probabilities that: `meth`=constraints, `meth`="FINISHED", `disc`="RESTART", and `disc`="REPEAT".

The informing information is represented as a six dimensional vector. The first value is the number of concepts where $p_{c1} > 0.8$. The second and third values are the probabilities for the most likely value of the from and to goals. In the case of the stops, the highest marginal probability for any road name is computed. The fourth, fifth and sixth values in the vector are binary flags denoting whether the departure point, arrival point and route exist in the database.

4.3. The hand-crafted policy

The initial BUDSLETSGO system uses a hand-crafted policy for deciding how to respond in the dialogue. All policy decisions are based on the summary state described above. First, the policy checks the discourse summary information and decides if it should repeat, restart or say goodbye. If the probability of the relevant node is above a threshold (0.8 for restart or goodbye, 0.5 for repeat) for each of these, then the relevant action is taken. If the probability for the restart value is above 0.5 then the system will confirm whether the user wants to restart.

If no discourse actions are taken then the system will check if further information is needed. The system will confirm the most likely value of any concepts where the closest point to the tuple (p_{c1}, p_{c2}) is one of $(0.7, 0.2)$, $(0.7, 0.0)$,

$(0.5, 0.4)$,, $(0.5, 0.2)$, $(0.5, 0.1)$, $(0.35, 0.35)$,. Otherwise the system will request, in order, where the user wants to leave from, where they want to go to, and what time they would like to leave.

5. OUTPUT GENERATION

The BUDSLETSGO system uses template-based rules to convert dialogue acts into word sequences. Each dialogue act can have multiple associated templates, and the system will choose randomly between them. These templates were based on the templates used in the Olympus-based system. Text-to-speech is implemented using the Baratinoo expressive speech synthesiser. Baratinoo is the industrial speech synthesiser developed at France Telecom, using state-of-the-art unit selection technology [8].

6. EVALUATION

6.1. Simulated evaluation

In order to evaluate the system during development, an agenda-based user simulator [9] was built for the Let's Go domain. The simulator models the user as having a goal and an agenda of planned user dialogue acts to perform in response to the system. The rules for adding and removing from the stack are probabilistic but hand-crafted.

Along with the user simulator, an error simulator was also built. The error simulator depends on an error rate parameter, r, an N-best list size (in these experiments $N = 3$), and a variability parameter, V (here $V = 32$). A confidence score parameter vector, $\boldsymbol{\alpha}$, of size $N + 1$ is defined by

$$\boldsymbol{\alpha}_r{}^\top = \left(Vr, \frac{V(1-r-r^2)}{N-1}, \ldots, \frac{V(1-r-r^2)}{N-1}, Vr^2 \right).$$

For each turn, a vector of confidence scores is drawn from the Dirichlet distribution with parameters $\boldsymbol{\alpha}_r$. These confidence scores are used as probabilities to draw a position between 1 and $N + 1$. The true user act is placed at this position in an $N + 1$-best list and all remaining positions are assigned a confused user act, using hand-crafted rules to alter it. The item at position $N + 1$ is dropped and the list is passed to the dialogue manager.

Figure 3 shows the results of the simulated evaluation. As expected, the performance decreases as errors increase. At low error rates, the system achieves close to 100% success, as one would hope. The reward is computed in turns of the number of turns until completion, T. For successful dialogues, the reward is computed as $20 - T$ and for unsuccessful dialogues the reward is $0 - T$.

6.2. Human evaluation

A human evaluation of the system was performed as part of the control tests for the Lets Go! spoken dialogue challenge.

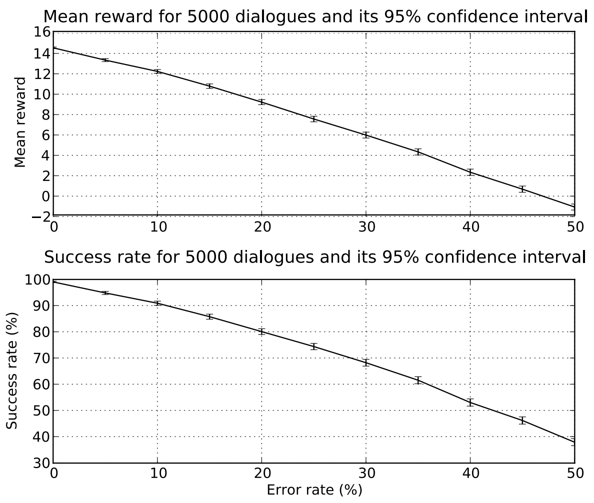

Fig. 3. The effect of errors on simulated performance for the BUDSLETSGO system.

A complete description of the challenge and the control tests is given in [10]. During the control tests, each site taking part in the challenge was asked to find test volunteers. These subjects were then asked to log on to on a web-site, and complete the tasks given to them there. In each task, the subject was given a place of departure, a place of arrival, a time of departure / arrival, and sometimes a bus route. Each subject was asked to complete two tasks with each of the four systems entered into the challenge.

After the control tests, all dialogues were sent to the organisers of the challenge. The organisers then transcribed all dialogues, attempted to match them against the web logs, and tagged them according to whether the system provided an acceptable bus route to the user. When a dialogue could not be matched to a web log it was ignored. If an acceptable bus route was provided, the dialogue is said to be successful.

In order to compute comparable word error rates (WERs) for the systems, all transcriptions were normalised. Three normalisations were performed:

- All words joined together in the dictionary were separated (e.g. A_M was separated to A M),
- All numbers were converted to word form (e.g. 64 was converted to "SIXTY FOUR"),
- Any utterances which were not in the transcriptions were ignored.

Table 2 provides the results of the challenge. The table shows that the success rate of the BUDSLETSGO system in the control tests was around 24.5% higher than the BASELINE system. Standard errors on the probability of success for the BUDSLETSGO and BASELINE systems are 3.6% and 5.0% respectively. This assumes a constant probability of success

System	# Dia.	# Suc.	% Suc.	WER
BASELINE	91	59	64.8 ± 5.0	42.38
SYSTEM2	61	23	37.7 ± 6.2	60.66
BUDSLETSGO	75	67	$\mathbf{89.3 \pm 3.6}$	**32.65**
SYSTEM4	83	62	74.7 ± 4.8	34.34

Table 2. Overall performance for the Lets Go! spoken dialogue challenge control tests. The columns show the system, number of dialogues (# Dia.), number of successful dialogues (# Suc), success rate (% Suc.) with one standard error, and overall word error rate (WER).

and independent dialogues. Under this assumption the number of successful dialogues follows a binomial distribution, and the difference in probability of success is statistically significant. The WER of the BUDSLETSGO system is around 9.7% lower than the BASELINE system in absolute terms.

Although the overall results show a good improvement in performance, it is difficult to establish how much of the improvement is due to speech recognition accuracy and how much is due to changes in the rest of the system. To differentiate between the two, a logistic regression of the probability of success against the word error rate for a particular dialogue was computed[1]. The results of this analysis are given in figure 4. For clarity, only the BUDSLETSGO and BASELINE systems are shown.

The figure gives an indication that the increase in performance obtained by the BUDSLETSGO system is due both to the improved word error rate and to increased dialogue management performance under high error rates. Most importantly, the rate of change in the success due to speech recognition errors is lower for the BUDSLETSGO system than for the BASELINE system. This corroborates the claim that POMDP systems are more effective at handling speech recognition errors.

A similar result can be obtained by grouping the data into WER bins. This alternative, however, is less sound theoretically as the performance for each bin will depend on the distribution of WERs within the bins. For this reason, only the results of the logistic regression are presented here.

7. CONCLUSION

This paper has described a spoken dialogue system for the Let's Go 2010 spoken dialogue challenge, built using Bayesian updates of dialogue state. The authors believe that the semantic decoder and output generation are comparable to the original Olympus system.

The first major difference between the BUDSLETSGO and BASELINE systems is the dialogue manager, which is based on the Bayesian updating framework instead of the agenda-based framework [11]. The Bayesian approach has the ad-

[1]The logistic regression uses the canonical logit link function.

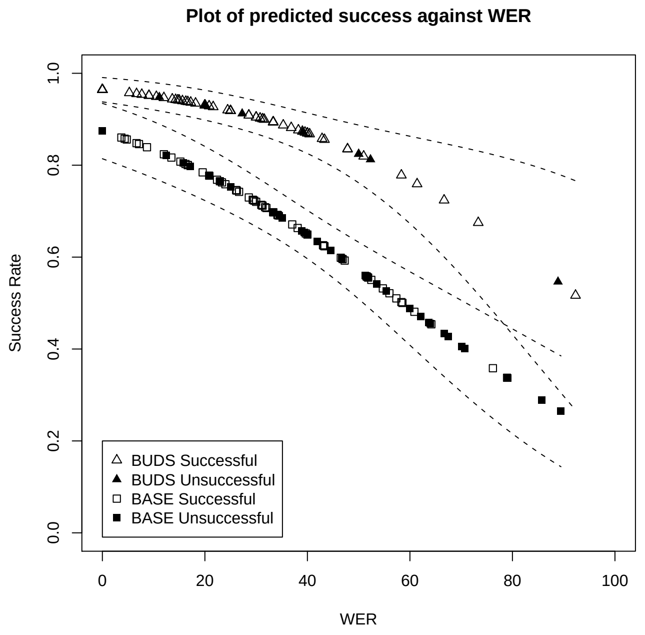

Fig. 4. The effect of WER on success rates of the BUDSLETSGO (BUDS) and BASELINE (BASE) systems for the Lets Go! control tests with human users. The plotted points show the true WER of a given dialogue along with the predicted probability of success according to the logistic regression model. Unfilled markers depict successful dialogues, while filled markers depict unsuccessful dialogues. The dotted lines show one standard error on each side of the predicted probability of success.

vantage that information from throughout the dialogue can be used to estimate the likelihood of any possible user goal.

The speech recogniser uses a universal language model, which is the other significant departure from the state-based approach used in the BASELINE system. The statistical language model has the potential for better modelling all possible user utterances as the model is trained from data. This is particularly true when the user strays from answering questions asked by the system. However, the use of a universal statistical language model does also enable the system to recognise words which may be unreasonable in a given context. If the N-best list of hypotheses is not long enough to include the true utterance, the POMDP user model will not be able to take it in to account.

Initial evaluations of the system indicate that the BUDSLETSGO system does outperform the BASELINE, as well as the other systems entered into the challenge [10]. Compared to the BASELINE system, the BUDSLETSGO system improves the dialogue success rate by 24.5% and the word error rate by 9.7%. A logistic regression of success against dialogue word error rate indicates that the improvement in success is due to both the speech recognition and the dialogue management. Note that these results are from the control tests. Live testing for the spoken dialogue challenge is currently under way.

In future work, the authors plan to investigate the effects of using a policy learned from reinforcement learning. A hand-crafted policy was used throughout the 2010 Lets Go! challenge. It is hoped that a learned policy will be able to further increase performance.

8. REFERENCES

[1] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The hidden information state model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, 2010.

[2] J. Williams, "Applying POMDPs to dialog systems in the troubleshooting domain," in *Proc HLT/NAACL Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*, 2007.

[3] U. Syed and J. D Williams, "Using automatically transcribed dialogs to learn user models in a spoken dialog system," *Proceedings of the ACL*, 2008.

[4] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech & Language*, 2010.

[5] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.0*, Cambridge University Press, 2000.

[6] A. Raux, D. Bohus, B. Langner, A. W Black, and M. Eskenazi, "Doing research on a deployed spoken dialogue system: One year of Let's Go! experience," in *Ninth International Conference on Spoken Language Processing*, 2006.

[7] B. Thomson, *Statistical methods for spoken dialogue management*, Ph.D. thesis, University of Cambridge, 2010.

[8] France Telecom, "Baratinoo expressive speech synthesiser," 2009, Demonstration can be downloaded from: http://tts.elibel.tm.fr.

[9] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young, "Agenda-based user simulation for bootstrapping a POMDP dialogue system," in *Proceedings of HLT/NAACL*, 2007.

[10] A. Black, S. Burger, B. Langer, G. Parent, and M Eskenazi, "Spoken dialogue challenge 2010," in *SLT Special Session: The spoken dialogue challenge*, 2010.

[11] D. Bohus and A. Rudnicky, "The RavenClaw dialog management framework: Architecture and systems," *Computer Speech & Language*, 2009.