Learning User Simulations for Information State Update Dialogue Systems

Kallirroi Georgila, James Henderson, Oliver Lemon

School of Informatics, University of Edinburgh, United Kingdom
{kgeorgil,jhender6,olemon}@inf.ed.ac.uk

Abstract

This paper describes and compares two methods for simulating user behaviour in spoken dialogue systems. User simulations are important for automatic dialogue strategy learning and the evaluation of competing strategies. Our methods are designed for use with "Information State Update" (ISU)-based dialogue systems. The first method is based on supervised learning using linear feature combination and a normalised exponential output function. The user is modelled as a stochastic process which selects user actions (⟨ speech act, task ⟩ pairs) based on features of the current dialogue state, which encodes the whole history of the dialogue. The second method uses n-grams of ⟨ speech act, task ⟩ pairs, restricting the length of the history considered by the order of the n-gram. Both models were trained and evaluated on a subset of the COMMUNICATOR corpus, to which we added annotations for user actions and Information States. The model based on linear feature combination has a perplexity of 2.08 whereas the best n-gram (4-gram) has a perplexity of 3.58. Each one of the user models ran against a system policy trained on the same corpus with a method similar to the one used for our linear feature combination model. The quality of the simulated dialogues produced was then measured as a function of the filled slots, confirmed slots, and number of actions performed by the system in each dialogue. In this experiment both the linear feature combination model and the best n-grams (5-gram and 4-gram) produced similar quality simulated dialogues.

1. Introduction

A central motivation for user simulation is to make it feasible to test the performance of different dialogue policies against simulated users in an efficient and inexpensive way. Using real users would require much more time and effort. In addition, every time we modified a dialogue strategy we would have to repeat all experiments with human users from scratch. Another powerful motivation for developing accurate user simulations is that they can be used to generate huge amounts of data, which we need for automatic learning of dialogue strategies (see e.g. [1, 3]). Models of dialogue state transitions that will feed Reinforcement Learning algorithms can be estimated either directly from a corpus or by generating training episodes using simulation. The advantage of the latter approach is that the system state representation need not be fixed: it can be changed and the system retrained without having to change the data set, data transcriptions or user model [3]. Several approaches to user simulation have been proposed (e.g. [2, 3]). Building on this work, we develop and compare two methods for simulating user behaviour. Our methods are designed for use with "Information State Update" (ISU) dialogue systems. The ISU approach to dialogue modelling supports the development of generic and flexible dialogue systems by using rich representations of dialogue context. Information states represent the state of a dialogue as a large set of features, e.g. speech acts, tasks, filled slots, ASR confidence score, etc. For full details see [4, 5].

This research is funded by the European Community FP6 Project "TALK", http://www.talk-project.org

Our first model of user behaviour is based on supervised learning using linear feature combination and a normalised exponential output function. Its output is a probability distribution over actions (⟨ speech act, task ⟩ pairs). The model is based only on features of the current dialogue state. Note that the current dialogue state does not encode only the last system action but also the whole history of the dialogue. The second method uses n-grams of ⟨ speech act, task ⟩ pairs restricting the length of the history taken into account up to the order of the n-gram. Both models simulate user behaviour at the intention level. Intentions represent the actual information that a dialogue participant wants to convey. An intention can be defined as the minimum piece of information that can be conveyed independently within a given application [3]. Current approaches to user behaviour modelling focus on the intention level, bypassing the speech and word levels. However, by incorporating performance statistics from other levels (such as speech recognition and natural language understanding), the resulting systems can simulate the performance of complete dialogue systems.

Both models were trained and evaluated on a subset of the COMMUNICATOR corpus, to which we have added annotations for user actions and Information States. We have built an environment in which we can run user simulations against a real system or a learnt strategy in order to generate dialogues. This environment has been implemented using DIPPER [5] available at http://www.ltg.ed.ac.uk/dipper. The DIPPER architecture is a collection of software agents for prototyping dialogue systems implemented on top of the Open Agent Architecture (OAA) [6]. DIPPER supports building (multimodal) dialogue systems, by offering a Dialogue Move Engine and interfaces to speech recognisers, speech synthesisers, parsers and other agents. The user simulations and our learnt system policies have been implemented as OAA agents in this environment.

In section 2 we discuss the annotations we have added to the COMMUNICATOR data. Section 3 describes our environment based on DIPPER and OAA as well as the 2 user simulation models based on n-grams and linear feature combination respectively. A brief description of the system policy is also given. In section 4 we present the tests carried out and finally in section 5 we end with conclusions and directions for future work.

2. COMMUNICATOR data annotation

The COMMUNICATOR corpora (2000 and 2001) consist of human-machine dialogues (approx 2300 dialogues in total) in the domain of flight reservation. The users always try to book

```
DIALOGUE LEVEL
Speaker: user
ConvDomain: [about_task]
SpeechAct: [provide_info]
AsrInput: <date_time>may eight</date_time>
TransInput: <date_time>may seventh</date_time>
Output:

TASK LEVEL
Task: [depart_date]
FilledSlot: [depart_date]
FilledSlotValue: [may eight]
GroundedSlot: [orig_city,dest_city]

LOW LEVEL
WordErrorRatenoins: 50.00
WordErrorRate: 50.00
SentenceErrorRate: 100.00
KeyWordErrorRate: 100.00

HISTORY LEVEL
SpeechActsHist: opening_closing,instruction,
  request_info,[provide_info],implicit_confirm,
  request_info,[provide_info],implicit_confirm,
  request_info,[provide_info]
TasksHist: meta_greeting_goodbye,meta_instruct,
  orig_city,[orig_city],orig_city,dest_city,[dest_city],
  orig_dest_city,depart_arrive_date,[depart_date]
FilledSlotsHist: [orig_city],[dest_city],[depart_date]
FilledSlotsValuesHist: [cincinnati],[denver],[may eight]
ConfirmedSlotsHist: [],[orig_city],[orig_city,dest_city]
```

Figure 1: *Example Information State (State 10 of a* COMMUNICATOR *dialogue). [User information]*

a flight, but they may also try to select a hotel or car-rental. The dialogues are primarily “slot-filling” dialogues, with some information being presented to the user. These corpora have been previously annotated (but only for the system’s side of the dialogues) using the DATE scheme’s Conversational Domain, Speech Act, and Task dimensions [7].

We used an automatic system to assign Speech Acts and Tasks to the user utterances, and to compute information states for each point in the dialogue (i.e. after every utterance) [8]. The system is implemented with DIPPER [5] and OAA [6], using several OAA agents. An example of some of the types of information recorded in information states is shown in figure 1. This is an information state corresponding to the user’s response after the system tried to implicitly confirm origin and destination cities, and then requested information about the departure date. As shown in the figure, the user provides information about the departure date, and also positively confirms (i.e. grounds) the origin city and destination city slots. The history level of the information states encodes information about the complete dialogue history.

For the experiments reported in this paper, we used the data from 4 of the 8 systems in the 2001 corpus. This subset consists of 97 users, 697 dialogues, and 51309 states.

3. The user and system simulations

The environment used for simulating both the system and the user employs DIPPER [5] and OAA [6] and can support various agents for system or user simulation. For the system simulation we can use either reinforcement learning, supervised learning with linear feature combination, n-grams, or hand-coded systems based on the ISU approach. For the user simulation we can use either supervised learning with linear feature combination or n-grams. This is due to the modularity of the simulation environment’s architecture, making it easy to incorporate other system or user agents. In the experiments described in the sequel we use supervised learning with linear feature combination for the system simulation, and either n-grams or supervised learning with linear feature combination for the user simulation. In the future we also plan experiments with a system based on n-grams and a baseline hand-crafted spoken dialogue system that is currently under development. Moreover, we have carried out experiments using a system policy learnt with hybrid reinforcement/supervised learning [9].

3.1. Simulating ASR errors

Ideally we would have dialogue data that contains automatic speech recognition (ASR) confidence scores. Unfortunately, the COMMUNICATOR corpus does not have this information, but it does contain both the output of the speech recognition engine for a user utterance and its manual transcription. By comparing the ASR output with the manual transcription we compute the word error rate (WER), which we assume is correlated with ASR confidence. During training the linear feature combination user model used information about the word error rate. Therefore in simulation mode DIPPER generates word error rates for each user action based on the distribution of word error rates in the COMMUNICATOR data. For about 70% of the user utterances the speech recognition performance was perfect (WER=0%), for 10% of the user utterances the speech recogniser produced a WER of 100% and for the rest of the user utterances WER values varied between 0 and 100%.

In the future we also intend to use keyword error rates (KER) since they are already included in the information states produced by the automatic annotation system. The KER shows the percentage of the correctly recognised keywords (cities, dates, times) and is computed after parsing the input utterances. It therefore simulates natural language understanding errors.

3.2. The n-gram user simulation

The user simulation based on n-grams treats a dialogue as a sequence of pairs of speech acts and tasks. It takes as input the n-1 most recent ⟨ speech act, task ⟩ pairs in the dialogue history, and uses the statistics of n-grams in the training set to decide on the next user action (one of the 48 ⟨ speech act, task ⟩ pairs) that will be passed to DIPPER. If no n-grams match the current history, the model can back-off to smaller n-grams. We use the annotated COMMUNICATOR data as a sequence of ⟨ speech act, task ⟩ pairs for the training data, and we use the CMU-Cambridge Statistical Language Modelling Toolkit v2 [10] to generate n-grams with multiple types of discounting, as described in section 4.

Although we currently only use it for user simulation, the n-gram simulation agent is parametric, so it can be used either for user simulation or for system simulation. During training, the n-gram agent picks only the n-grams which are appropriate for the type of model, namely the n-grams which end in system/user actions for the system/user simulations respectively. During simulations, the n-gram user model chooses the next action according to the distribution of n-grams, while the n-gram system model chooses the single action with the best score. The system can perform multiple actions at the same turn, while the user can only perform one action. In preliminary tests with a n-gram based system policy, backing-off led to seemingly random dialogues, so currently backing off is permitted only for the n-gram based user simulation and the first action of the system (to ensure that the system always produces at least one action before it releases the turn).

3.3. Simulations based on linear feature combination

The ISU framework is significantly different from the frameworks used in previous work on learning dialogue management policies for user behaviour, in that the number of possible states is extremely large. Having a large number of states is a more realistic scenario for practical, flexible, and generic dialogue systems, but it also makes many learning approaches intractable. To overcome the large state space we need to exploit commonalities between different states. The feature-based nature of ISU state representations expresses exactly these commonalities between states through the features that the states share. There are a number of techniques that can be used for learning with feature-based representations of states, but the simplest and most efficient is to use a linear combination of features.

We use a linear combination of features to map from a vector of real valued features $f(s)$ for the state s to a probability distribution $\hat{P}(a|s)$ over user actions a. The trained parameters of the linear function are a vector of weights w_a for each action a. Given weights trained on a given dataset, an estimate $\hat{P}(a|s)$ of the probability of the user doing action a given state s is the normalised exponential function applied to the inner product of the state vector $f(s)$ and the weight vector w_a[1].

$$\hat{P}(a|s) = \frac{\exp(f(s)^T w_a)}{\sum_a \exp(f(s)^T w_a)} \qquad (1)$$

The weights w_a are trained on the state-action pairs observed in the training data. For training, we use a simple gradient descent learning method, with weight decay regularisation.

The mapping $f(s)$ from states to vectors must be specified before learning. Each value in these vectors represents a possible commonality between states, so it is through the definition of $f(s)$ that we control the notion of commonality which will be used by the linear function. The definition of $f(s)$ we are currently using is a straightforward mapping from feature-value pairs in the information state s to values in the vector $f(s)$. One area of future research is to investigate more complicated mappings $f(s)$, such as the application of kernel methods.

The state vector mapping $f(s)$ is computed using the first four levels of our annotation of the COMMUNICATOR data. We chose a subset of the annotations and converted them into one of three types of features. For annotations which take numbers as values, we used a simple function to map these numbers to a real number between 0 and 1, with the absence of any value being mapped to 0. For annotations which can have arbitrary text as their values, we used 1 to represent the presence of text and 0 to represent no value. The remaining annotations all have either a finite set of possible values, or a list of such values. Annotations with list values are first converted to lists of pairs consisting of the annotation label and each value. For every possible label-value pair, we define a feature in the vector $f(s)$ which is 1 if that label-value pair is present in the state and 0 if it is not. These form the vast majority of our 290 features.

The dialogue system model which we use in our evaluation of user models is also based on a linear combination of state features, and the same set of state features from the annotated COMMUNICATOR data. Supervised learning is used to train the linear model to predict the next system action (out of a total of 70 actions), and the most probable next action is taken as the system policy's choice – see [9] for full details.

[1]We will use the notation $x^T y$ to denote the inner product between vectors x and y (i.e. "x transpose times y"). This form of model is sometimes called a maximum entropy model, and is equivalent to a single layer neural network probability estimator.

4. Experimental results

First we divided the COMMUNICATOR corpus in training and test sets and computed the perplexity of our models. Then we evaluated our user simulation models by running them against a learnt system policy. Both the user simulations and the system policy were trained using the annotated COMMUNICATOR data for the ATT, BBN, CMU, and SRI systems. The quality of the simulated dialogues produced was then measured as a function of the filled slots, confirmed slots, and number of actions performed by the system in each dialogue.

4.1. Perplexity estimation

We used a 10-fold cross-validation technique, computing perplexity for each of the test subsets and then calculating the average perplexity across all 10 trials. We computed perplexity for the n-grams ($2 \leq n \leq 5$) using 4 different types of discounting: absolute, linear, Good-Turing, and Witten-Bell. For the user model based on linear feature combination, we used only absolute discounting. The results are given in table 1.

Model	Discounting	Perplexity	Std Dev
linear	absolute	2.08	0.20
5-gram	absolute	4.01	0.71
	linear	4.36	0.88
	Good-Turing	3.93	0.69
	Witten-Bell	3.61	0.52
4-gram	absolute	3.83	0.57
	linear	4.16	0.64
	Good-Turing	3.79	0.57
	Witten-Bell	3.58	0.45
3-gram	absolute	4.03	0.53
	linear	4.23	0.58
	Good-Turing	4.07	0.56
	Witten-Bell	3.94	0.46
2-gram	absolute	4.93	0.57
	linear	4.96	0.58
	Good-Turing	4.99	0.61
	Witten-Bell	4.94	0.55

Table 1: *Average perplexity and standard deviations (10 folds): both user models, several discounting schemes.*

The linear feature combination model has the lowest perplexity, as expected, since it uses more features than the n-grams and the complete dialogue history. Interestingly the 4-gram has lower perplexities than the 5-gram. We hypothesise that this could be due to overtraining for the 5-gram (it produced very low perplexities when tested on the training data).

4.2. Evaluation against the system policy

We also evaluated the different user simulations by running them against a learnt system policy. For these experiments, we restrict our attention to users who only want single-leg flight bookings. Thus the user simulations were restricted not to produce actions related to continuation. This was done for 2 reasons. First, it is easier to evaluate the generated dialogues if all users have the same goals. In our case, there are only 4 essential slots to be filled: origin city, destination city, departure date, and departure time. If we also allowed continuation trips then it would be difficult to count the correct number of slots to be filled since it is not known in advance how many legs the trip will have. Secondly, user goals were not encoded in the

COMMUNICATOR corpus, so training users with goals would be complicated. The importance of incorporating user goals in a user model has been shown in previous work [3], so in future work we will build different user models according to the user goal, or alternatively add features relevant to user goals in the information state representation.

To evaluate the success of a dialogue, we use the final state of the dialogue to compute a scoring function. Because currently we are considering users who only want single-leg flight bookings, the scoring function looks at the four slots relevant to these bookings: origin city, destination city, departure date, and departure time. We give 25 points for each slot which is filled, plus another 25 points for each slot which is also positively confirmed (i.e. grounded). We also deduct 1 point for each action performed by the system, to penalise longer dialogues. Thus the maximum possible score is 198 (i.e. 200 minus 2 system actions: ask for all the user information in one turn, and then offer a flight). The motivation behind this evaluation metric is that confirmed slots are more likely to be correct than slots which are just filled. If we view the score as proportional to the probability that a slot is filled correctly, then this scoring assumes that confirmed slots are twice as likely to be correct.

During testing, each user simulation was run for 1000 dialogues against the linear feature combination system policy. The final state for each one of these dialogues was then fed through the scoring function and averaged across dialogues. The results are shown in table 2. There are 5 different user simulations: linear feature combination, bigram, trigram, 4-gram, and 5-gram. When we run the user simulations against a system policy we do not use discounting as we did for calculating perplexity because we want the user model to produce only known actions. However, we needed discounting in the experiments of section 4.1 because it could be the case that actions that did not exist in the training set appeared in the test set.

Model	Total score	Filled slots	Confirmed slots	Length penalty
linear	109.8	85.3	48.0	-23.5
5-gram	108.9	72.2	51.1	-14.4
4-gram	107.2	73.0	49.1	-14.9
3-gram	101.4	68.7	48.2	-15.5
2-gram	71.2	71.3	29.2	-29.3

Table 2: *Average rewards across 1000 simulated dialogues.*

The linear, 5-gram, and 4-gram models all produced similar scores. After estimating the distribution of system turn lengths in the COMMUNICATOR corpus it was shown that the 5-gram and 4-gram models cover 93.91% and 88.39%, respectively, of the sequences of contiguous system actions appearing in the data. The trigram and the bigram models cover only 66.08% and 36.86%, which explains their worse performance.

5. Conclusion

We described and compared two methods for simulating users of spoken dialogue systems. Our methods are designed for use with "Information State Update" (ISU) dialogue systems. The first method is based on supervised learning using linear feature combination and a normalised exponential output function. The user is modelled as a stochastic process which selects user actions ($\langle$speech act, task$\rangle$ pairs) based on features of the current dialogue state, which encodes the whole history of the dialogue. The second method uses n-grams of $\langle$speech act, task$\rangle$ pairs, restricting the length of the history taken into account by the order of the n-gram. Both models were trained and evaluated on a subset of the COMMUNICATOR corpus, to which we have added annotations for user actions and Information States. The model based on linear feature combination gave the best perplexity, followed by the 4-gram. Each one of the user models ran against a system policy trained on the same corpus with a method similar to the one used for our linear feature combination model. The quality of the simulated dialogues produced was then measured as a function of the filled slots, confirmed slots, and number of actions performed by the system in each dialogue. In this experiment, both the linear feature combination model and the best n-grams (5-gram and 4-gram) produced similar results.

Our future work will focus on incorporating user goals in user simulations either by building different user simulations according to the user goals or alternatively by adding features relevant to user goals in the information state representations. Moreover, we will perform further experiments using different system policies i.e. policies developed using Reinforcement Learning or produced by hand-crafted baseline systems.

6. References

[1] Levin, E. and Pieraccini, R., "A stochastic model of computer-human interaction for learning dialogue strategies", Proc. Eurospeech, pp. 1883-1886, 1997.

[2] Eckert, W., Levin, E., and Pieraccini, R., "User modelling for spoken dialogue system evaluation", Proc. IEEE ASR Workshop, 1997.

[3] Scheffler, K. and Young, S.J., "Corpus-based dialogue simulation for automatic strategy learning and evaluation", Proc. NAACL Workshop on Adaptation in Dialogue Systems, pp. 64-70, 2001.

[4] Larsson, S. and Traum, D., "Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit", Natural Language Engineering, 6(3-4), 2000.

[5] Bos, J., Klein, E., Lemon, O., and Oka, T. "DIPPER: Description and formalisation of an Information-State Update dialogue system architecture", Proc. 4th SIGdial Workshop on Discourse and Dialogue, pp. 115-124, 2003.

[6] Cheyer, A. and Martin, D. "The Open Agent Architecture", Journal of Autonomous Agents and Multi-Agent Systems, 4(1/2):143-148, 2001.

[7] Walker, M.A., Passonneau, R.J., and Boland, J.E., "Quantitative and qualitative evaluation of Darpa Communicator spoken dialogue systems", Proc. ACL, pp. 515-522, 2001.

[8] Georgila, K., Lemon, O., and Henderson, J., "Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations", to appear in DIALOR, 2005.

[9] Henderson, J., Lemon, O., and Georgila, K., "Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data", submitted to 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, 2005.

[10] Clarkson, P.R. and Rosenfeld, R., "Statistical language modeling using the CMU-Cambridge toolkit", Proc. Eurospeech, 1997.