User Simulation for Spoken Dialogue Systems: Learning and Evaluation

Kallirroi Georgila, James Henderson, Oliver Lemon

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

Abstract

We propose the "advanced" n-grams as a new technique for simulating user behaviour in spoken dialogue systems, and we compare it with two methods used in our prior work, i.e. linear feature combination and "normal" n-grams. All methods operate on the intention level and can incorporate speech recognition and understanding errors. In the linear feature combination model user actions (lists of $\langle$ speech act, task $\rangle$ pairs) are selected, based on features of the current dialogue state which encodes the whole history of the dialogue. The user simulation based on "normal" n-grams treats a dialogue as a sequence of lists of $\langle$ speech act, task $\rangle$ pairs. Here the length of the history considered is restricted by the order of the n-gram. The "advanced" n-grams are a variation of the normal n-grams, where user actions are conditioned not only on speech acts and tasks but also on the current status of the tasks, i.e. whether the information needed by the application (in our case flight booking) has been provided and confirmed by the user. This captures elements of goal-directed user behaviour. All models were trained and evaluated on the COMMUNICATOR corpus, to which we added annotations for user actions and dialogue context. We then evaluate how closely the synthetic responses resemble the real user responses by comparing the user response generated by each user simulation model in a given dialogue context (taken from the annotated corpus) with the actual user response. We propose the expected accuracy, expected precision, and expected recall evaluation metrics as opposed to standard precision and recall used in prior work. We also discuss why they are more appropriate metrics for evaluating user simulation models compared to their standard counterparts. The advanced n-grams produce higher scores than the normal n-grams for small values of n, which proves their strength when little amount of data is available to train larger n-grams. The linear model produces the best expected accuracy but with respect to expected precision and expected recall it is outperformed by the large n-grams even though it is trained using more information. As a task-based evaluation, we also run each of the user simulation models against a system policy trained on the same corpus. Here the linear feature combination model outperforms the other methods and the advanced n-grams outperform the normal n-grams for all values of n, which again shows their potential. We also calculate the perplexity of the different user models.

Index Terms: dialogue systems, user simulation, learning and evaluation of dialogue strategies, evaluation metrics

1. Introduction

User simulations are becoming increasingly important in the field of dialogue systems due to their use in automatic dialogue strategy learning and the evaluation of competing strategies. Machine-learning approaches to dialogue management have been proposed by several authors [3, 7, 9]. These techniques focus on learning optimal dialogue strategies from data and/or simulations. They use Reinforcement Learning (RL) to find a policy which optimises a reward function over dialogues. However, it is rarely the case that enough training data is available to sufficiently explore the vast space of possible dialogue states and strategies. Therefore, a promising approach is to use small corpora to train stochastic models for simulating real user behaviour. Once such a model is available, any number of dialogues can be generated through interaction between the simulated user and the dialogue policy, and can be used for RL. Another 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 the dialogue strategy we would have to repeat all experiments with human users from scratch.

User simulation can be on the intention level (e.g. speech act, task – provide_information, destination_city) or on the surface level (words comprising utterances). Current approaches to user behaviour modelling for learning focus on the intention level. However, by incorporating performance statistics from other levels (such as speech recognition and natural language understanding), the resulting systems can simulate the performance of input processing of whole dialogue systems. User simulation on the surface level is mostly useful for evaluation purposes and can also be used to accelerate the development process of dialogue systems [6]. Several approaches to user simulation on the intention level have been proposed [4, 9], but none of them uses the rich contextual information that we employ here.

Note that this paper is an extension of our work in [1]. Here we use a much more detailed annotation of the COMMUNICATOR corpus and new methods for both training and evaluating user models. We introduce the "advanced" n-grams and the expected accuracy, expected precision, and expected recall metrics as opposed to standard precision and recall introduced in prior work [4]. Our methods are designed for use with "Information State Update" (ISU) dialogue systems that use rich representations of context [8], but can also be used for systems with simpler definitions of context.

In section 2 we discuss the annotations we have added to the COMMUNICATOR data. Section 3 describes our user simulation models based on linear feature combination and n-grams. A brief description of the system policy is also given. In section 4 we present our evaluation metrics. In section 5 we describe the experiments carried out and discuss the results. Section 6 presents our conclusions and directions for future work.

2. The COMMUNICATOR data set

Our data set is the COMMUNICATOR corpus (dialogues in the domain of flight, hotel, and car reservations) annotated as a sequence of information states (IS) recording dialogue context phenomena such as grounding and user intentions [2]. An example IS is shown in figure 1. Due to space restrictions not all IS fields are depicted.

Note the group of IS fields { PreviouslyFilledSlots, PreviouslyFilledSlotsValues, PreviouslyConfirmedSlots }, which inform us about the current status of the slots and thus may only contain one instance per slot. We shall use this information in the advanced n-grams method, where we condition users' actions not only on their previous speech acts and tasks but also on whether slots associated with the current task have been filled or confirmed.

For the experiments reported in this paper, we used the complete COMMUNICATOR 2001 corpus. This subset consists of 8 systems, 202 users, 1683 dialogues, and 125,388 states. In [1, 3, 4] a preliminary version of these annotations has been used (4 systems, 97 users, 697 dialogues, and 51,309 states). In addition, the annotations were not as detailed as the current ones.

```
DIALOGUE LEVEL
Speaker: user
ConvDomain: [about_task]
SpeechAct: [provide_info,provide_info]
AsrInput: <date_time>may eight morning</date_time>
TransInput: <date_time>may seventh morning</date_time>
Output:
TASK LEVEL
Task: [depart_date,depart_time]
FilledSlot: [depart_date,depart_time]
FilledSlotValue: [may eight,morning]
GroundedSlot: [orig_city,dest_city]
LOW LEVEL
WordErrorRate: 33.33
KeyWordErrorRate: 50.00
HISTORY LEVEL
SpeechActsHist: opening_closing,instruction,
  request_info,[provide_info],implicit_confirm,
  request_info,[provide_info],implicit_confirm,
  request_info,[provide_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,depart_time]
PreviouslyFilledSlots: [orig_city],[dest_city],
  [depart_date],[depart_time]
PreviouslyFilledSlotsValues: [cincinnati],[denver],
  [may eight],[morning]
PreviouslyConfirmedSlots: [orig_city],[dest_city]
```

Figure 1: *An example Information State. User information appears between [] parentheses.*

3. The user and system simulations

3.1. Simulating ASR and understanding errors

A notable constraint on data to be useful for machine learning is that all captured features should in principle be available to a dialogue system at runtime – so that a dialogue system using a learnt policy can compute a next action in any state. That means that in [2] we needed to annotate the automatic speech recognition (ASR) hypotheses of the systems, rather than the transcribed user utterances that are also provided in the corpus. Moreover, a parser was incorporated in our automatic annotation system to extract the semantic information of those utterances. Therefore the data we use for training our user models (both linear feature combination and n-grams) incorporates both ASR and natural language understanding (NLU) errors. In the future we intend to extend our annotations of the users' turns based on the transcriptions of user utterances and then apply techniques for simulating ASR and NLU errors that do not depend on specific ASR and parsing modules. Now our models simulate the ASR errors of the speech recognisers in the COMMUNICATOR systems and the NLU errors of our parser.

3.2. The linear feature combination user simulation model

The ISU framework is significantly different from the frameworks used in previous research on learning dialogue 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 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 state vector mapping $f(s)$ is computed using the four levels of our annotation of the COMMUNICATOR data (see figure 1 and section 2), that is, the model is trained using the complete information state and not only information about the speech act and task. There are 1282 features. The linear user model will choose the next user action (out of a total of 522 lists of $\langle$ speech act, task $\rangle$ pairs) according to a distribution learnt from the data. More details are given in [1, 3]. Note that the linear feature combination model described in [1, 3] could produce only a single $\langle$ speech act, task $\rangle$ pair instead of a list.

The dialogue system policy which we use in our evaluation of user models is also based on a linear combination of state features from the annotated COMMUNICATOR data. Supervised learning is used to train the linear feature combination model to predict the next system action, and the most probable next action is taken as the system policy's choice – see [3] for full details. The actions are $\langle$ speech act, task $\rangle$ pairs. In addition, there are *release_turn* and *end_dialogue* actions. There are a total of 74 actions which occur in the data. The set of state features is the same as those used for the linear feature combination user simulation model.

3.3. The "normal" n-gram user simulation model

The user simulation based on n-grams treats a dialogue as a sequence of lists of $\langle$ speech act, task $\rangle$ pairs. It takes as input the n-1 most recent lists of $\langle$ speech act, task $\rangle$ 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 522 lists of $\langle$ speech act, task $\rangle$ pairs). 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 lists of $\langle$ speech act, task $\rangle$ pairs for training data, and we use the CMU-Cambridge Statistical Language Modelling Toolkit v2 to generate n-grams using absolute discounting for smoothing the n-gram probabilities.

An example 3-gram that would lead to the user action shown in figure 1 would be:
{$\langle$ system, implicit_confirm, orig_dest_city $\rangle$, $\langle$ system, request_info, depart_date $\rangle$, $\langle$ user, [(provide_info, depart_date), (provide_info, depart_time)] $\rangle$}.

N-gram models have the advantage of being purely probabilistic, fully domain-independent, and can be trained easily once enough data is available. Their weakness is that they may not place enough constraints on the user to simulate realistic behaviour. The generated responses may correspond well to the previous system action, but often they do not make sense in the wider context of the dialogue (for example, failing to incorporate goal-directed user behaviour). One solution to this problem is to use a high order of n. However, n cannot be arbitrarily high due to data sparsity issues.

3.4. The "advanced" n-gram user simulation model

In order to provide n-grams with more relevant information about the context of the dialogue we built "advanced" n-gram models where we condition not only on the speech acts and tasks but also on the status of the associated slots, i.e. whether they are filled and confirmed. The idea behind this approach is that the user is not very likely to provide information again about a slot that has been confirmed but rather will proceed to fill the remaining slots in order to accomplish his/her task.

For this reason we add "status flags" to the n-grams, and in this case the 3-gram that would lead to the user action shown in figure 1 would now become:
{ $\langle$ system, implicit_confirm, orig_dest_city, 0 $\rangle$, $\langle$ system, request_info, depart_date, 0 $\rangle$, $\langle$ user, [(provide_info, depart_date), (provide_info, depart_time)], 0 $\rangle$} .

The status flag can be either 1 if all associated slots are both filled and confirmed or 0 in all other cases. In the previous example when the system implicitly confirmed the "orig_dest_city" (2 states before the state of figure 1) none of the "orig_city" and "dest_city" were both filled and confirmed (they were only filled), hence the status flag was 0. In the same way when the system requested information about "depart_date" (1 state before the state of figure 1) the status flag was 0 since the "depart_date" slot was not filled or confirmed.

This approach incorporates some aspect of goal-directed user behaviour. The results in section 5 indicate that it produces better user simulation models than the normal n-grams.

4. Evaluation metrics

There are no generally accepted criteria as to what constitutes a good user simulation model in dialogue systems. In our view, a good user model should be able to generate "human-like" behaviour and dialogues that make sense while interacting with a system policy. We must also remember that user simulations are to be used for automatic dialogue strategy learning. For this reason we desire user simulations which do not always act in the same way in identical states – we need to explore the policy space, so some reasonable amount of variation is required. All our user simulation models are stochastic, i.e. they produce actions based on a probability distribution learnt from the training data and not always the action with the highest probability. Our metrics have been calculated based on this distribution.

4.1. Accuracy, precision, recall, and perplexity

To assess whether our models produce human-like behaviour we need to compare their output with real responses given by users in the same contexts. For this purpose we propose the expected accuracy, expected precision, and expected recall metrics.

We define expected accuracy (EA) as the expected proportion of simulated user turns that match exactly the real user turn. In other words, the EA is the percentage of the time that the model would choose the same action as the observed action if we ran the model for a large number of times through the same states found in the data, each time generating an action according to the distribution produced by the model. This measure is also known as "predicted probability" [10]. One consequence of this measure is that the maximum EA is not 1. For example, if in a particular context 90% of real users supply origin city information, and the remainder ask for help, then the best EA we can hope for is 90% (always choosing to supply the origin city). Also, if the model has the same behaviour as real users, then the EA is even lower, at $0.9 * 0.9 + 0.1 * 0.1 = 82\%$.

We use expected precision (EP) and expected recall (ER) to quantify how closely the synthetic turn resembles the real user turn. Precision and recall are a common measure of "goodness" in user modelling [10] and they were first used in evaluating user simulation models in dialogue systems by [4]. In [4] precision and recall were calculated always choosing the action with the highest probability. In the n-gram examples of sections 3.3 and 3.4 we can see that it is possible for an action to have more than 1 components, i.e. the action [(provide_info, depart_date), (provide_info, depart_time)] consists of the 2 action components (provide_info, depart_date) and (provide_info, depart_time). As with accuracy, we measure the expected value of precision and recall we would get if we selected from the models distribution a large number of times. EP measures the proportion of correct action components among all the predicted action components. An action component is considered correct if it matches at least one of the action components in the real user response. ER measures how many of the action components in the real response are predicted correctly.

Perplexity (PP) is widely used for measuring the performance of language models but we believe it can also be a useful metric for the evaluation of user simulation models. We first introduced it as a measure of the quality of a user simulation model in [1]. One way to interpret PP is as a measure of the number of possible actions that the user model has to choose between at a given state.

4.2. Task performance

We evaluated our user simulation models by running them against a learnt system policy (see section 3.2). 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. We give 25 points for each slot which is filled, plus another 25 points for each slot which is also positively confirmed. We also deduct 1 point for each system action performed, to penalise longer dialogues. The maximum possible score is 198 (i.e. 200 minus 2 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. This scoring assumes that confirmed slots are twice as likely to be correct.

5. Experimental results

We divide the COMMUNICATOR corpus in training and test sets. For all our experiments we use 2-fold cross validation.

5.1. Comparing simulated and real user responses

The expected accuracy results for all models are displayed in figure 2. The large n-grams produce the best expected precision (EP) and expected recall (ER). That means that the large n-grams produce the best performance on the action component level. This is a bit surprising, since the linear model has access to more information to decide on its action. However, the linear model produces the highest expected accuracy (EA), i.e. performs better on the whole action level. The two types of n-grams (normal and advanced) have similar EA, EP and ER values for 5-grams and 4-grams. However, the advanced 3-grams and 2-grams have a better performance than their normal counterparts, which means that the additional information they have about the status of the filled and confirmed slots can be an advantage. Unfortunately the 5-grams and 4-grams suffer from data sparsity which becomes more severe for their advanced version and that explains the lack of improvement. Nevertheless considering that the smaller n-grams are the ones that cannot capture much information about the history and therefore need information from additional features to be robust, our result is promising. We must also keep in mind that it is difficult to collect enough data to produce robust n-grams with a high value of n. The strength of the advanced n-grams is that they are better than the normal n-grams for smaller values of n, i.e. values of n that are used in practice to produce robust estimates.

In [4] the best standard precision and recall scores were reported using the Pietquin model [9] and led to precision of 40.16% and recall of 33.38%. However, these results appear to be calculated with the model always choosing its best action, which does not correspond to the real behaviour of the model, and is likely to produce better results than the EP and ER. For example, when we use this way of measuring precision and recall for the linear model, we get 55.18% precision and 52.65% recall, which are much better results than those for EP and ER. We also get an accuracy of 52.12% for the linear model when the best action is always chosen. These results are also much better than the above results from [4]. However, we cannot compare these results directly due to the fact that all models in [4] were trained using a preliminary version of the data which we use here (only 4 systems with less detailed annotations). Nevertheless it is interesting that we produce better or similar results even though none of our models explicitly incorporates goals, as is the case for the Pietquin model, and our models are trained and tested on very detailed data with higher perplexity and more actions to choose between. [10] used both accuracy and "predicted probability" (equivalent to our EA) to compare the performance of different predictive models. Their results showed that EA provides finer-grained information than accuracy.

The perplexity (PP) results are depicted in figure 2. The linear feature combination model has the lowest PP, as expected, since it uses more features than the n-grams and the complete dialogue history. PP is related to EA because they both take into account the probability distribution of actions. However, PP is more severely affected by assigning a very low probability for the observed action, such as for an unseen event. These cases will add a large value to cross-entropy. This explains why the advanced n-grams always have higher PP than the their normal counterparts for all values of n whereas, as it was explained above, the advanced 3-grams and 2-grams produce higher EA than their normal versions.

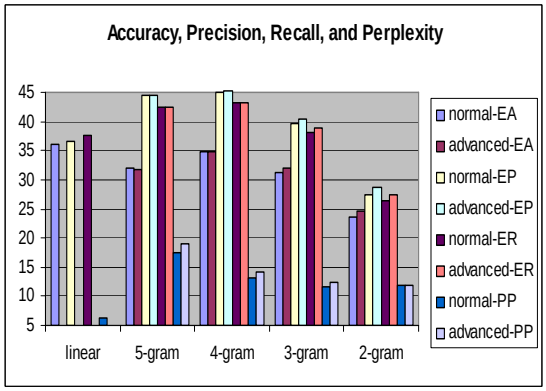

Figure 2: *Expected accuracy (% EA), expected precision (% EP), expected recall (% ER), and perplexity (PP) results.*

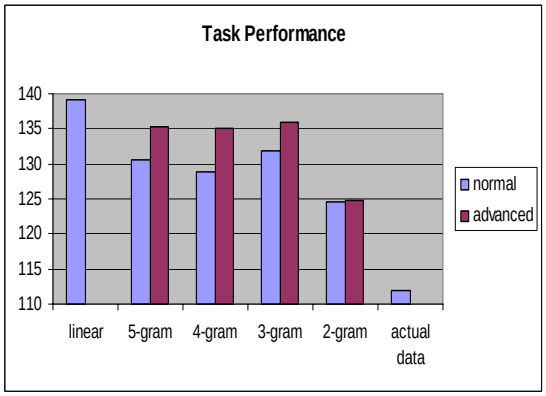

Figure 3: *Results for different user simulations evaluated against a learnt system policy.*

5.2. Evaluation against a learnt system policy

We also evaluated the different user simulations by running them against a learnt system policy. 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.

Each user simulation was run for 1000 dialogues against the system policy. The final state for each one of these dialogues was then fed through the scoring function described in 4.2 and averaged across dialogues. The results are shown in figure 3. The baseline score is produced by feeding the actual data through the scoring function. The linear model produces better scores than the n-grams and the advanced n-grams are consistently better than the normal n-grams. This is true also for 5-grams and 4-grams even though for these values of n the advanced n-grams suffer from data sparsity and the normal n-grams have similar expected accuracy and lower perplexity. It seems that the advanced 5-grams and 4-grams suffer from data sparsity but when they have estimates these are robust. Higher order n-grams (both normal and advanced) suffer from data sparsity and back-off very often to 3-grams. That explains the similar task performance of 5-grams, 4-grams and 3-grams. After estimating the distribution of system turn lengths in the COMMUNICATOR corpus it was shown that the 5-gram, 4-gram and 3-gram models cover 92.56%, 88.15% and 71.61% respectively, of the sequences of contiguous system actions appearing in the data. That means that the 3-gram has adequate coverage of the sequences of system actions and explains once more the good performance of 3-grams. On the other hand the coverage of bigrams is only 42.08%. From all the metrics we use, perplexity and expected accuracy are the ones that reflect the results in task completion, i.e. the linear model that has the lowest perplexity and highest expected accuracy produces the best performance in terms of filled slots, confirmed slots and length penalty.

6. Conclusions

We described and compared three methods for simulating users of spoken dialogue systems and three evaluation metrics for measuring their performance. We discussed why expected accuracy, expected precision and expected recall are more appropriate metrics for user simulation than their standard counterparts since they are based on the probability distribution of actions and not only on the action with the highest probability.

The strength of the advanced n-grams is that they outperform normal n-grams especially for small values of n. This is very important because it is rarely the case that the training data is adequate to generate robust n-grams with high values of n. The advanced n-grams prove to be a good trade-off between the linear feature combination model which requires the complete information state for training and the normal n-grams that are trained using limited information.

Our future work will focus on using other methods (e.g. Bayesian networks) to model the user more accurately. Bayesian networks are designed to capture more complex dependencies and will better model the reasoning of the user. We have used the linear model to train a system policy [3] which we then evaluated with real users against a hand-crafted policy [5]. Preliminary results showed that the learnt policy produced a gain of 14.2% in perceived task completion. We intend to use also the advanced n-grams for training policies and then evaluate them with real users. This is the best way to evaluate user simulations with respect to how good are in practice the system policies that they train.

7. References

[1] Georgila, K., Henderson, J., and Lemon, O., "Learning User Simulations for Information State Update Dialogue Systems", Proc. EUROSPEECH-INTERSPEECH, 2005.

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

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

[4] Schatzmann, J., Georgila, K., and Young, S., "Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems", Proc. SIGdial, 2005.

[5] Lemon, O., Georgila, K., Henderson, J., and Stuttle, M., "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system", Proc. EACL (demo session), 2006.

[6] Chung, G., "Developing a flexible spoken dialog system using simulation", Proc. ACL, 2004.

[7] Singh, S., Litman, D., Kearns, M., Walker, M., "Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System", Journal of Artificial Intelligence Research (JAIR), 2002.

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

[9] Pietquin, O., "A framework for unsupervised learning of dialogue strategies", PhD thesis, Fac. Polytechn. Mons, 2004.

[10] Zukerman, I. and Albrecht, D., "Predictive Statistical Models for User Modeling", User Modeling and User-Adapted Interaction, 2001.