INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems

Layla El Asri, Jing He, Kaheer Suleman

Maluuba Research
Montreal, Canada
first.last@maluuba.com

Abstract

User simulation is essential for generating enough data to train a statistical spoken dialogue system. Previous models for user simulation suffer from several drawbacks, such as the inability to take dialogue history into account, the need of rigid structure to ensure coherent user behaviour, heavy dependence on a specific domain, the inability to output several user intentions during one dialogue turn, or the requirement of a summarized action space for tractability. This paper introduces a data-driven user simulator based on an encoder-decoder recurrent neural network. The model takes as input a sequence of dialogue contexts and outputs a sequence of dialogue acts corresponding to user intentions. The dialogue contexts include information about the machine acts and the status of the user goal. We show on the Dialogue State Tracking Challenge 2 (DSTC2) dataset that the sequence-to-sequence model outperforms an agenda-based simulator and an n-gram simulator, according to F-score. Furthermore, we show how this model can be used on the original action space and thereby models user behaviour with finer granularity.

Index Terms: spoken dialogue systems, user simulation, dialogue management

1. Introduction

Statistical Spoken Dialogue Systems (SDS) [1, 2, 3] typically require several thousands of dialogues to learn a good dialogue strategy [4, 5]. It is costly to collect this quantity of dialogues; therefore, research has turned to user simulation [6, 7, 8, 9]. A user simulator is expected to have the following properties: to be statistically consistent with real users, to generate coherent sequences of actions, and to generalize to new contexts [10]. User simulation can be either at the intention level, *i.e.*, generating dialogue acts [11, 9], or at the utterance level [12]. In this work, we focus on the intention level.

Many models have been designed in order to meet the requirements cited above [6, 13, 14, 15, 11, 9]. These models typically suffer from important drawbacks, which include the inability to take dialogue history into account [6], the need of rigid structure to ensure coherent user behaviour [16], heavy dependence on a specific domain [13], the inability to output several user intentions during one dialogue turn [12], or the requirement of a summarized action space for tractability [11].

In this paper, we introduce a sequence-to-sequence model for user simulation. The simulator is modelled with an encoder Recurrent Neural Network (RNN) and a decoder RNN [17]. The encoder takes as input the entire history of the dialogue, encoded as a sequence of dialogue contexts. It outputs an internal representation of this sequence. This representation is passed as input to the decoder. The decoder generates a sequence of dialogue acts corresponding to user intentions.

We train this model on the Dialogue State Tracking Challenge 2 (DSTC2) [18] dataset. This corpus consists of dialogues between real users and an SDS in the domain of restaurant-seeking. We compare the sequence-to-sequence model to an agenda-based simulator [16, 11], an n-gram simulator, and a sequence-to-one RNN which takes the same input as the sequence-to-sequence model but chooses an output from among a list of predefined sequences of acts. We show that the RNN-based models outperform the other two simulators on the F-score measure. We also show on the DSTC3 dataset [19] that the RNN-based models generalize best to new domains.

In the next section, we discuss previous models for user simulation. Then, in Section 3, we describe the sequence-to-sequence model. Section 4 presents the DSTC2 and DSTC3 corpora, the models and the metrics used for comparison, and the results of our experiments.

2. Background

User simulation, at the intention level, consists of predicting the next user dialogue act depending on the dialogue history and the user goal. The first user simulator was proposed by Eckert et al. [6] who used a simple bi-gram model $P(a_u|a_m)$ to predict the next user act a_u given the last system act a_m. This model does not produce coherent behaviours from the user because the user only reacts to the latest system action. This issue can be overcome by restricting the types of actions that the user can draw from according to dialogue history, which requires more engineering effort. Scheffler and Young [13] proposed a graph-based model. Therein, all possible paths for user behaviour are mapped into a network. The main difficulty of this approach is that it requires extensive domain knowledge and engineering. Pietquin and Dutoit [20] suggested a Bayesian model for user behaviour. They added an explicit representation of the user goal and memory to the probabilistic bi-gram model. The user's action was then conditioned on her goal and memory. Georgila et al. [14] proposed a richer model of the user with the information state approach [21]. The information state carries information on the current state, the dialogue history and ongoing actions. The authors investigated learning user behaviour by using a 4-gram representation and a linear combination to map each state to a vector of features. Cuayáhuitl et al. [15] used a hidden Markov model (HMM) for user simulation. The model generated both user and system actions. Schatzmann et al. [16] proposed a new agenda-based approach that did not necessarily need training data but could be trained in case such data was available [11]. Chandramohan et al. [9] proposed to model the

http://dx.doi.org/10.21437/Interspeech.2016-1175

Machine output / User answer	Machine acts	Inconsistency vector	Constraints status	Requests status
Welcome! How may I help you?	0000000010 greet	000000	001	10110111
Is there a cheap restaurant downtown?				
A cheese restaurant. What is your budget?	0000010001 implicit-confirm, request	000010	011	10110111
No, I said a cheap restaurant.				
Panda express is a cheap restaurant downtown.	0100000100 offer, inform	000000	111	10110111
What is the address of this place?				
Panda express is located at 108 Queen street.	0100000100 offer, inform	000000	111	10111111

Table 1: Examples of contexts in a dialogue with a restaurant-seeking system. The user goal has two constraints (cheap and downtown) and one request (address).

user as a decision-making agent and to model user behaviour with reinforcement learning.

An important feature for user simulation, which encourages coherent behaviour throughout a dialogue, is the ability to take into account the dialogue history. For tractability reasons, previous models do not account for a long dialogue history. Another important consideration is that users who interact with SDS often utter several dialogue acts during a single dialogue turn. This feature is not often represented in user simulators, as it would quickly become inefficient to compute a model with an output space containing all possible sequences of dialogue acts. To deal with this, in the agenda-based approach, a stack-like structure is added to the model to provide a coherent set of dialogue acts that can be output at the same time. In the next section, we propose a model that takes into account the entire dialogue history and outputs a sequence of dialogue acts without relying on any external structure.

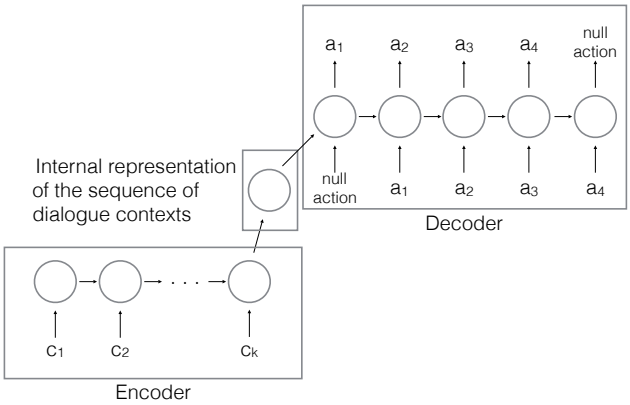

Figure 1: Sequence-to-sequence model for user simulation.

3. The Sequence-to-sequence Model

Figure 1 represents the sequence-to-sequence user simulation model. The model takes as input a sequence of dialogue contexts $(c_1, c_2, ..., c_k)$ and outputs a sequence of actions $(a_1, a_2, ..., a_l)$.

Similarly to Schatzmann et al. [16, 11], at the beginning of each dialogue, we uniformly draw a goal $G = (C, R)$ where C is a set of *constraints* and R is a set of *requests*. For a restaurant-seeking system, constraints are typically expressed over the type of food, the price range, and the area where the restaurant is located. Requests can include these slots as well as the restaurant's name, its address, its phone number, *etc*.

A context c_t at turn t is defined by the following components:

- the most recent machine acts $a_{m,t}$,
- the inconsistency between the most recent information provided by the machine and the user goal inconsist_t,
- the constraints status (informed or not) const_t, and
- the requests status (informed or not) req_t.

The machine acts are encoded as a vector $a_{m,t}$ of size n_{ma}[1]. The vector $a_{m,t}$ has ones for the current machine acts and zeros everywhere else. The inconsistency is composed of two vectors whose size equals the number of possible constraints n_c. Both vectors are initialized at 0 and reset after each turn. After the machine makes a proposition to the user (*e.g.*, a proposition of restaurant), all of the constraint slots which are in the user goal but which were not mentioned by the machine are set to 1 in the first vector. Every time the machine mentions a slot provided by the user (*e.g.*, in a confirmation or a proposition), all of the constraint slots which have been misunderstood are set to 1 in the second vector. The inconsistency vector is thus a turn-level vector which models the system's understanding of the user goal. The constraints status vector is of size n_c and keeps track of what the user has said to the machine. The constraints which are not in C are set to 1 and those in C are set to 0. Every time the user provides a constraint to the SDS, this constraint is set to 1. A constraint is reset to 0 every time it is set to 1 in the inconsistency vector or if the machine requests this slot. The requests status vector is of size n_r where n_r is the number of possible requests. This vector has ones for all slots which are not in the user goal and zeros for the slots in R. A request slot is set to 1 every time the SDS mentions it in a proposition. The requests status vector is reset after each new proposition from the system. Examples of updates are given in Table 1. At time t, the sequence-to-sequence model takes as input the entire sequence of contexts that have been observed

[1] where n_{ma} is the number of possible machine acts.

so far, which models dialogue history. This input is passed to an RNN which outputs a single vector v_t corresponding to the model's internal representation of dialogue history. The encoder and the decoder have similar structures. They are both based on a Long Short-Term Memory (LSTM) [23]. In both cases, the LSTM is followed by a fully connected layer. The input of the encoder is a sequence of contexts c_t:

$$c_t = a_{m,t} \odot inconsist_t \odot const_t \odot req_t,$$

where $\odot$ is concatenation. The LSTM are implemented following these equations:

$$\begin{aligned} i_t &= \sigma(W_i c_t + U_i h_{t-1}) \\ f_t &= \sigma(W_f c_t + U_f h_{t-1}) \\ C_t &= i_t * \tanh(W_c c_t + U_c h_{t-1}) + f_t * C_{t-1} \\ o_t &= \sigma(W_o x_t + U_o h_{t-1}) \\ h_t &= o_t * \tanh(C_t), \end{aligned} \quad (1)$$

where i_t is the input gate, σ is the sigmoid function, f_t is the forget gate, o_t is the output gate, C_t is the cell gate and h_t is the hidden state. The last output of the LSTM is passed to one layer fully connected which outputs v_t. Then, v_t is used to initialize the decoder LSTM at each time step [22]. During training, the decoder is fed with ground truth, *i.e.*, the sequences of user acts observed in the dataset given the history of contexts. During runtime, the only input to the decoder is the *null* action. The decoder is implemented according to the same equations as the encoder. It is followed by *softmax* activation in order to compute a distribution of probabilities over the actions. The first action $a_{t,1}$ is drawn according to the output distribution of the first step of the LSTM. Then it is fed as input to the second step. This process is repeated until a sequence of l actions $(a_{t,1}, ..., a_{t,l})$ (including one or more *null* actions at the end of the sequence) has been generated. We train this model with a categorical cross-entropy loss function.

Each sequence output by the simulator is a sequence of dialogue acts, *e.g.*, `(inform, request)`. We map these dialogue acts to actions such as `inform(type of food = Chinese), request(price range)` by looking at the current user goal and uniformly drawing among the constraints left to inform and the requests left to ask. In the case of a confirmation asked by the system or if the system misunderstood a slot, we map the `inform` dialogue act to the slot in question. We show in the following section that it is also possible to train the model on original actions directly, *e.g.*, `request-area`, which removes this post-processing step and models user behaviour at a finer level.

4. Experiments

In this section, we compare the sequence-to-sequence simulator to an agenda-based simulator, a sequence-to-one model, and an n-gram model. We train these models on the training set of DSTC2.

We define a *user compound act* $\tilde{a}_t^u$ as a sequence of dialogue acts $(a_{t,1}^u, ..., a_{t,l}^u)$, where $l \geq 1$. All the models compared in this section output user compound acts. Similarly, we define machine compound acts as $\tilde{a}_t^m$.

4.1. User Simulation Models

The first baseline for comparison is a simple bi-gram model, which outputs a compound act $\tilde{a}_t^u$ given the last machine compound act $\tilde{a}_t^m$. We compute probabilities for the 54 possible user compound acts in the DSTC2 dataset.

In the agenda-based model, the user is modelled with a pair (G, A), where G is the goal and A is the agenda. As explained in Section 3, the goal is a pair (C, R), where C is a set of constraints and R is a set of requests. The agenda A is a stack-like structure which contains all of the `inform` and `request` acts needed by the user in order to perform her goal.[2] At each dialogue turn t, the user simulator samples a single act a_t^u based on the current dialogue context d_t[3]. Then, based on the chosen act a_t^u, the user simulator samples the number n of acts to pop from the stack. The compound act $\tilde{a}_t^u$ is then formed by a_t^u and the acts that are popped from the stack. The dialogue context d_t does not only include the latest dialogue acts spoken by the system, it also includes information on the dialogue history. For instance, if the SDS proposes a restaurant to the user and, in another dialogue turn, answers one of the user's requests regarding this restaurant, d_t will include an indication over the goal status for this restaurant. The dialogue contexts combined with the agenda guarantee coherent user behaviour throughout the dialogue. This feature, as well as the fact that the model outputs one or several dialogue acts at each turn, makes this a good model for comparison with the sequence-to-sequence approach.

The third simulator is a sequence-to-one model. This model takes the same input as the sequence-to-sequence model but only outputs a probability distribution over a predefined set of compound acts. This set of size 54 contains all of the compound acts in DSTC2.

4.2. F-score

We compare the 4 models based on F-score. The F-score is the geometric mean of the precision and the recall, which are computed as follows:

$$\text{precision} = \frac{\text{number of correctly predicted dialogue acts}}{\text{number of predicted dialogue acts}}$$

$$\text{recall} = \frac{\text{number of correctly predicted dialogue acts}}{\text{number of dialogue acts in the corpus}}$$

$$\text{F-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.$$

4.3. The DSTC2 dataset

DSTC2 is a publicly available dataset composed of a training set of 1612 dialogues, a validation set of 506 dialogues and a test set of 1117 dialogues. The training and validation sets were collected with two handcrafted policy managers whereas a statistical policy manager was used for the test set. The dialogues were collected with real users who had been given a goal consisting of a set of constraints and a set of requests. Each user interacted with the system in order to find a restaurant matching all of the constraints and then to collect the information in the requests. The user dialogue acts tagged in this dataset are as follows: `deny`, `null` (empty act), `request more`, `confirm`, `acknowledge`, `affirm`, `request`, `inform`, `thank`, `repeat`, `request alternative` (ask for an-

[2] If the user is looking for an Indian restaurant downtown and wants to know the price range, the agenda will be: `inform(food = Indian)`, `inform(area = downtown)`, `request(price range)`.

[3] Since the dialogue contexts are not expressed in the same way for the sequence-to-sequence model and the agenda-based model, we use different notations.

Dataset	Bigram	Agenda-based	Sequence-to-one	Sequence-to-sequence
DSTC2 Validation	0.20	0.24	0.37	0.34
DSTC2 Test	0.09	0.18	0.29	0.27
DSTC3 Test	—	0.13	0.19	0.18

Table 2: Average F-score on 50 runs.

other option), `negate`, `goodbye`, `hello` and `restart` (ask the system to restart the dialogue).

This dataset offers an interesting setting since we can use both the validation and test sets in order to evaluate the user simulators. In general, a user simulator is designed for a given policy manager: data is collected with this manager then the user simulator model is trained on this data and evaluated with the same policy manager. With this dataset, we have the possibility to follow this methodology (on the validation set) but we are also able to evaluate on a set of dialogues on the same domain but collected with a different policy manager (the test set). Therefore, we can evaluate the extent to which each model captures the behaviour of real users in unseen settings for the same task.

4.4. Results

Table 2 presents results on the validation and test sets of DSTC2. The first observation is that, as expected, the bi-gram model performs relatively poorly. On both the validation and test sets, the RNN-based models significantly outperform the agenda-based model in terms of F-score. The sequence-to-one model performs slightly better than the sequence-to-sequence model because it is a simpler problem to learn a distribution over a given set of sequences than to output each sequence step by step. However, the sequence-to-sequence model performs very closely to the sequence-to-one simulator, demonstrating that this model can achieve good performance. In addition, a considerable advantage of this model concerns scalability. In particular, the number of possible compound acts might grow considerably if the sets of constraint and request slots were of larger size and/or if the number of dialogue acts was larger. The output space would rapidly become too large for training the sequence-to-one model on a small dataset and it would likely be more efficient to use the sequence-to-sequence model. A further advantage is that the sequence-to-sequence model can be used on the original act space.

We illustrated this property with a second experiment, in which we modify the sequence-to-sequence model to train it on the original action space. The dialogue acts generated by the simulator are uniformly mapped to the user goal as discussed in Section 3. In this experiment, we circumvent this random mapping by increasing the number of possible acts. Instead of having one `inform` dialogue act, we define three separate acts: `inform_food`, `inform_pricerange`, `inform_area`. The advantage of this format is that a mapper is no longer needed and users can be modelled at finer granularity. Indeed, as shown in Table 3, it is possible to learn the order in which constraint and request slots are provided to the system by users. For instance, in the case that the user goal includes food, area and price range, the encoder-decoder model learns, in proportions commensurate with those found in the corpus, that the food slot is most often preferred as the first slot (72% in the corpus, 48% for the simulator), then the price range (16% vs. 31%), and then the area (12% vs. 21%).

Slot	in goal	corpus	sequence-to-sequence
area	yes	6	23.1
price range	no	0	10.4
food	yes	140	221.0
area	yes	15	101.8
price range	yes	19	153.1
food	yes	86	238.2

Table 3: For two different user goals, we compute the count of when a slot has been the first to be provided to the system in the corpus and by the sequence-to-sequence model (averaged over 10 runs). Note that when the user informs the system of a slot which is not in the goal, the value for this slot is *do not care*.

The last experiment involves evaluating the simulators on the DSTC3 test set [19]. DSTC3 is a dataset of 2264 dialogues with a system that can search for restaurants, pubs and coffee shops. Compared to DSTC2, in this dataset, the number of possible constraints is increased with the following slots: children allowed, has internet, has tv, near (*e.g.*, nearby Queens college) and type (restaurant, pub or coffee shop). The user and system dialogue acts can easily be mapped to those in DSTC2. We use this dataset in order to evaluate the user simulators on a new, larger domain. We train the models on DSTC2 as before, and evaluate them on the DSTC3 test set based on F-score. The results are presented in Table 2. These show that the sequence-to-one and sequence-to-sequence models significantly outperform the agenda-based model. Compared to DSTC2, there is a degradation in F-score which can be explained by the fact that this new domain has a larger set of compound acts (we found 40 compound acts which never occurred in DSTC2). The degradation concerns mostly the recall. Notably, the F-score for these models is similar to the F-score of the agenda-based model on the test set of DSTC2.

5. Conclusions

We proposed a new sequence-to-sequence model for user simulation in spoken dialogue systems. Compared to previous models, this simulator takes into account the entire dialogue history, it does not rely on any external data structure to ensure coherent user behaviour, and it does not require mapping to a summarized action space, which makes it able to model user behaviour with finer granularity. We showed that this model outperforms a state-of-the-art simulator based on the F-score measure. We also showed that it can be efficiently transferred to a new information-seeking domain. In future work, we will use the model to train a statistical spoken dialogue system and further explore the potential of this architecture.

6. References

[1] E. Levin, R. Pieraccini, and W. Eckert, "Learning dialogue strategies within the markov decision process framework," in *Proc. of IEEE ASRU*, 1997.

[2] M. Gašić, F. Jurčíček, B. Thomson, K. Yu, and S. Young, "On-line policy optimisation of spoken dialogue systems via live interaction with human subjects," in *Proc. of IEEE ASRU*, 2011.

[3] L. Daubigney, M. Geist, S. Chandramohan, and O. Pietquin, "A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimisation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 8, pp. 891–902, 2012.

[4] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-efficient batch reinforcement learning for dialogue management optimization," *ACM Transaction on Speech and Language Processing*, vol. 7, no. 3, pp. 1–21, 2011.

[5] M. Gašić, M. Henderson, B. Thomson, P. Tsiakoulis, and S. Young, "Policy optimisation of pomdp-based dialogue systems without state space compression," in *Proc. of SLT*, 2012.

[6] W. Eckert, E. Levin, and R. Pieraccini, "User modeling for spoken dialogue system evaluation," in *Proc. of IEEE ASRU*, 1997, pp. 80–87.

[7] K. Georgila, J. Henderson, and O. Lemon, "User simulation for spoken dialogue systems: Learning and evaluation," in *Proc. of Interspeech*, 2006.

[8] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies," *The Knowledge Engineering Review*, vol. 21, no. 2, 2006.

[9] S. Chandramohan, M. Geist, F. Lefèvre, and O. Pietquin, "User simulation in dialogue systems using inverse reinforcement learning," in *Proc. of Interspeech*, 2011.

[10] O. Pietquin and H. Hastie, "A survey on metrics for the evaluation of user simulations," *Knowledge Engineering Review*, vol. 28, no. 01, pp. 59–73, 2013.

[11] J. Schatzmann, B. Thomson, and S. Young, "Statistical user simulation with a hidden agenda," in *Proc. of SIGDIAL*, 2007.

[12] S. Jung, C. Lee, K. Kim, M. Jeong, and G. G. Lee, "Data-driven user simulation for automated evaluation of spoken dialog systems," *Computer Speech and Language*, vol. 23, no. 4, pp. 479–509, 2009.

[13] K. Scheffler and S. J. Young, "Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning," in *Proc. of HLT*, 2002, pp. 12–18.

[14] K. Georgila, J. Henderson, and O. Lemon, "Learning user simulations for information state update dialogue systems," in *Proc. of Eurospeech*, 2005.

[15] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Human-computer dialogue simulation using hidden markov models," in *Proc. of ASRU*, 2005, pp. 290–295.

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

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, 2014.

[18] M. Henderson, B. Thomson, and J. Williams, "The Second Dialog State Tracking Challenge," in *Proceedings of SIGDIAL*, 2014.

[19] ——, "The Third Dialog State Tracking Challenge," in *Proc. of IEEE SLT*, 2014.

[20] O. Pietquin and T. Dutoit, "A Probabilistic Framework for Dialog Simulation and Optimal Strategy Learning," *IEEE Transactions on Audio, Speech and Language*, vol. 14, no. 2, pp. 589–599, 2006.

[21] S. Larsson and D. Traum, "Information state and dialogue management in the trindi dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, pp. 323–340, 2000.

[22] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, 2014.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] V. Rieser and O. Lemon, "Cluster-based user simulations for learning dialogue strategies," in *INTERSPEECH*, 2006.