Combining User Intention and Error Modeling for Statistical Dialog Simulators

Silvia Quarteroni[1], *Meritxell González*[1,2], *Giuseppe Riccardi*[1], *Sebastian Varges*[1]

[1] DISI - University of Trento, 38050 Povo (Trento), Italy
[2] TALP Center, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain

silviaq@disi.unitn.it, mgonzalez@lsi.upc.edu, riccardi@disi.unitn.it, varges@disi.unitn.it

Abstract

Statistical user simulation is an efficient and effective way to train and evaluate the performance of a (spoken) dialog system. In this paper, we design and evaluate a modular data-driven dialog simulator where we decouple the "intentional" component of the User Simulator from the Error Simulator representing different types of ASR/SLU noisy channel distortion. While the former is composed by a Dialog Act Model, a Concept Model and a User Model, the latter is centered around an Error Model. We test different Dialog Act Models and Error Models against a baseline dialog manager and compare results with real dialogs obtained using the same dialog manager. On the grounds of dialog act, task and concept accuracy, our results show that 1) data-driven Dialog Act Models achieve good accuracy with respect to real user behavior and 2) data-driven Error Models make task completion times and rates closer to real data.

1. Introduction

Data-driven techniques are a widely used approach to the development of robust (spoken) dialog systems, particularly when training statistical dialog managers (DMs) [1, 2]. Generating the data to train such DMs can be costly as potential users are not always available for the task at hand; moreover, once the data is available, it must be manually analyzed and annotated. This is why user simulators (US) have been introduced to replace real conversations with synthetic ones and optimize a number of SDS components. Indeed, several approaches exist to the design of user simulators, as illustrated in [1]: as we aim to train statistical DMs [2], we focus on the *intention* (rather than lexical) level of simulation, as formalized in [3].

In this paper, we: 1) design a simulator where the Error Model derives its parameter estimates from real conversations; 2) define and implement different simulation models, by varying the Dialog Act Model and the Error Model components; 3) evaluate different simulators against real dialogs on the grounds of dialog act, task and concept accuracy. In particular, Section 2 presents our simulator architecture, Section 3 presents the simulation environment where we conduct our experiments, illustrated in Section 4. Finally, Section 5 positions our research in the context of related work and our conclusions and future work are summarized in Section 6.

2. Simulator Architecture

Data-driven simulation takes place within the rule-based version of the ADASearch system [2], which uses up to 16 dialog acts (described in [4]) to deal with three tasks and a dozen concepts related to lodging and events in Trentino, Italy. Since simulation in our framework occurs at *intention level*, the simulator and DM exchange *actions*, i.e. ordered sequences of *dialog acts* and (optionally) *concept-value* pairs. As illustrated in Figure 1, at turn t, the DM issues an action a_s, defined as an ordered dialog act sequence $a_s = \{da_0, .., da_n\}$, where each dialog act carries zero or more concept-value pairs: $da_j = da_j(c_0(v_0), .., c_m(v_m))$. For instance, we could have $a_s = \{$Apology(); Clarif-request(Event_type(fair))$\}$.

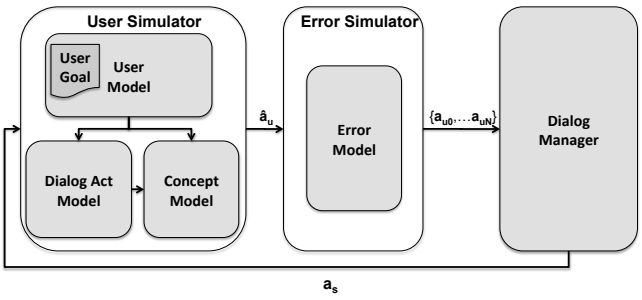

Figure 1: Architecture of the simulation environment

A User Simulator and an Error Simulator are then involved: the former estimates a plausible user action $\hat{a}_u$ given the DM action a_s; the latter distorts $\hat{a}_u$ into an N-best list of simulated actions $S = \{a_u^0, .., a_u^N\}$ received by the DM at $t + 1$ in "replacement" of the user-ASR-SLU pipeline. A confidence score is associated with each simulated action a_u^j and each individual concept forming a brick of interpretation.

In order to generate the list S of upcoming action hypotheses, the probability of each action being generated after the previous DM action a_s is estimated based on the conversation context. Such a context is represented by a User Model, a Dialog Act Model, a Concept Model and an Error Model. In particular,

- the **User Model** simulates the behavior of an individual user in terms of goals and other caller-specific features such as cooperativeness and tendency to hang up;
- the **Dialog Act Model** generates a distribution of M actions $A_u = \{a_u^0, .., a_u^M\}$, each a_u^i being a plausible sequence of dialog acts and concepts given a_s; one action $\hat{a}_u$ is chosen out of A_u following specific criteria;
- the **Concept Model** generates concept values for $\hat{a}_u$ by estimating $P(c_i(v_i)|da(c_0, .., c_m))\ \forall i \in [0..m]$;
- the **Error Model** simulates the noisy ASR-SLU channel by "distorting" $\hat{a}_u$ with errors; it estimates

Work partly funded by EU project ADAMACH (contract 022593).

$P(c_i(\hat{v}_i)|c_i(\tilde{v}_i))$, $\forall c_i$, where $\hat{v}_i$ may differ from the intended value for c_i, $\tilde{v}_i$. Eventually, the N-best list of noisy interpretations $S = \{a_u^0, .., a_u^N\}$ is returned.

In this modular architecture, the different levels and phases of simulation are factored out, allowing a plug-and-play comparison of simulation regimes as illustrated in Sections 2.1–2.4.

2.1. User Model

The User Model represents user-specific features, both persistent (cooperativeness, patience, no input probability) and transient, i.e. the current user goal. In particular, cooperativeness is a real value representing the ratio of concepts mentioned in a_s that also appear in $\hat{a}_u$; this ratio is computed turn-by-turn from a collection of real conversations and then averaged to a value *coop* (see [3]). Patience is defined as the tendency to abandon the conversation (hang up event), resulting in $pat = P(HangUp|a_s)$. Similarly, no input probability accounts for user behavior in noisy environments, resulting in $noi = P(NoInput|a_s)$. Finally, the user goal UG is represented as a task name and the list of concepts and values required to fulfill it: an example of UG is {Activity(EventEnquiry), Time_day(2), Time_month(may), Event_type(fair), Location_name(Povo)}.

As our experiments aim at comparing different Dialog Act and Error Models with the same persistent User Model features, we fix *coop*, pat and noi to their mean values as found in our data, while UG varies from dialog to dialog following goal distribution in the same data. As soon as the DM action is received by the simulator, it is passed to the User Model which generates a $HangUp$ with probability pat; else, a $NoInput$ is returned to the DM in place of action hypotheses at probability noi; otherwise, UG and *coop* are propagated to the following models.

2.2. Dialog Act Model

We define two Dialog Act (DA) Models, named Obedient and Task-based. In the Obedient (OB) model, infinite patience and cooperativeness are assumed of the user, who will always respond to each query requiring values for a set of concepts with an answer concerning exactly such concepts. Formally, the model responds to a DM action a_s with a single user action $\hat{a}_u$, obtained by consulting a rule table, having probability 1. In case a request for clarification is issued by the DM, this model returns a clarifying answer. Any offer from the DM to continue the conversation will be either readily met with a new task request or denied at a fixed probability. The OB model is useful to represent an "ideal", predictable user behavior; in [2], a RL-based dialog manager has been trained with this model.

The Task-based (TB) model is a variation of the bigram model defined in [5]. Here, a matrix records the transition frequencies of system actions to user actions, including hang up and no input/no match. Given a DM action a_s, the model responds with A_u, a list of M user actions whose probabilities derive from action distribution in real data:

$$A_u = \{(a_u^0, P(a_u^0|a_s)), .., (a_u^M, P(a_u^M|a_s))\}.$$

As one of the main drawbacks of the bigram model is that the space of user actions includes incompatible actions with respect to the user goal, the TB model (formalized as goal model in [6], but not experimented with using real data) only takes into account the actions taken under a specific task T_k as annotated in the training data. The TB model divides the training data into one partition for each T_k, then creates a bigram model for each partition, by computing:

$$A_u = \{(a_u^0, P(a_u^0|a_s, T_k)), .., (a_u^M, P(a_u^M|a_s, T_k))\}, \ \forall\, k.$$

In order to vary the simulation behavior, the choice of $\hat{a}_u$ out of A_u is a random sampling from A_u according to the distribution of probabilities therein. As the partition of the training data reduces the number of observations, the TB model includes a strategy to back off to the simpler bigram model and even to the unigram distribution of DAs as a last resort.

2.3. Concept Model

The Concept Model takes the action selected by the DA Model, $\hat{a}_u = \{da_0, .., da_n\}$, and attaches values and interpretation confidences to dialog act concepts. In this work, the Concept Model assigns the corresponding User Goal values for the required concepts: hence, $P(c_i(v_i)|da(c_0, .., c_n)) = 1$ if $c_i(v_i) \in UG$, 0 otherwise; in case $c_i \notin UG$, c_i will not appear in the final action. Similar to [7], the confidence attached by the CM to a $c_i(v_i)$ interpretation is randomly chosen within the average±std.dev. range of such concept as observed in the available data when the concept is correct. The Error Model may modify such confidences, as described below.

2.4. Error Model

The Error Model is responsible of simulating the noisy communication channel between user and system; as we simulate the error at SLU level, errors consist of incorrect concept values.

We have been experimenting with a data-driven (DD) model by which the precision Pr_c obtained by a concept c in a real dataset is used to estimate the frequency with which an error in the true value $\tilde{v}$ of c will be introduced: $P(c(v)|c(\tilde{v})) = (1 - Pr_c)$. For specific concepts, such as the intended task, a finer error scheme based on a confusion matrix of concept values has been adopted; this was not the case for concepts for which the amount of data was insufficient. The confidence attached to an erroneous interpretation is modified to a random value in the average±std.dev. of such concept when incorrect. The Error Model outputs the N-best[1] list $S = \{a_u^0, .., a_u^N\}$. For the sake of comparison, we also implement a "truthful" Error Model (TRU), representing noise-free communication, by which the top action in S coincides with the correct interpretation and has confidence 1.0.

3. Simulation Environment

The dialog manager used in both real and simulated dialogs follows an Information State Update approach [8], where the information state is represented as a database shared by all modules of the SDS [2]. The dialog state contains (simulated) SLU results, application information already provided by the (simulated) user including their grounding status, and counts of $NoInput$ events, amongst others. Given this information, the DM employs a 'dialog move engine' to determine the system action and response: this uses several sets of forward chaining inference rules. Typical dialog moves available to the system are those that are needed for the application domain, for example forward looking moves such as 'question-parameter', 'confirm-parameter', and 'request-repeat', and backward looking moves such as 'accept-parameter-implicitly' (by the system) or 'answer-question-parameter' (by the user). The DM initially

[1] To align with the behavior of our DM [2], $N = 2$ in our experiments.

issues action [Greet();Offer()], open to a wide range of user utterances, then dialog becomes more constrained.

An example of DM - simulator action exchange from real data is illustrated in Figure 2: each action is an ordered sequence of *dialog acts* and *concept-value pairs* with their confidence.

Figure 2: An example of Simulator (SIM) - DM exchange

```
DM: [Greet();Offer()]
SIM: [Info-request(activity=EventEnquiry{0.65};
   type=expo{0.65})]
DM: [Info-request(location)]
SIM: [Answer(location=Vela{0.6})]
DM: [Info-request(month)]
SIM: [Answer(month=November{0.23})]
DM: [ClarificationRequest(month= November)]
SIM: [Yes-answer({0.82})]
DM: [Info-request(day)]
SIM: [Answer(day=7{0.88})]
DM: [ReportOnAction();Inform(dblookup);Offer()]
SIM: [Info-request(activity=EventEnquiry{0.64})]
DM: ...
```

4. Evaluation

We evaluate our simulator models using two sets of metrics: first, "offline" metrics are used to assess how realistic the action estimations by DA Models are with respect to training data (Sec. 4.1). Then, "online" metrics (Sec. 4.2) evaluate end-to-end simulator performance by comparing real dialogs with fresh simulated dialogs in terms of dialog act distributions, error robustness and task duration and completion rates.

4.1. "Offline" metrics

In order to compare simulated and real user actions, we perform an evaluation of dialog act Precision (P) and Recall (R). Following [3], these are measured in a turn-by-turn basis as: $P = \frac{\#correct\ DA}{\#DA\ simulated\ action}$; $R = \frac{\#correct\ DA}{\#DA\ real\ action}$.

The unit of comparison is the dialog act and the set of attached concepts, as it is the action composition that the DA Models produce. We consider that a simulated dialog act is correct when it appears in the real action, including its concepts. We use 5-fold cross-validation on the set of 74 dialogs obtained from the ADASearch system (see Sec. 3). Each dataset consists of a sequence of turns containing the representation of the DM and the user actions. For each DM action a_s, the DA Model responds with A_u, the list of user actions and their probabilities as described in Sec. 2.2. The simulator then chooses a user action $\hat{a}_u$ from A_u, to be compared with the real user choice $\tilde{a}_u$.

Table 1: Precision and Recall compared to real user behavior

Model	Simulation		Most likely	
	P	R	P	R
Obedient	33.8	33.4	33.9	33.5
Task-based	**44.4**	**44.0**	**51.1**	**50.8**

Table 1 (col. "Simulation") shows P/R obtained for the OB and the TB models, clearly illustrating that the latter is much better at reproducing real action selection. As a reference, we also compare the most likely user action a_u^* as found in the simulator training data with the real user choice (Table 1, col. "Most likely"): a_u^* improves the expected P/R, providing a sort of "oracle" performance for the DA Model. However, as pointed out in [9] the purpose of simulation is to take into account not only the "mean" user behavior but *any possible* behavior, hence action sampling as illustrated in Sec. 2.2 is preferable. Nevertheless, the real test concerns online deployment of the simulators with different user behaviors, such as "fresh" user goals and data, as discussed in the next section.

4.2. "Online" metrics

We compare dialogs generated in the simulation environment using the same DM and different combinations of DA and Error Models on the grounds of dialog act distribution, concept P/R, and task completion rate/time. In order to align with our dataset of about 60 real dialogs, we ran the same amount of simulated dialogs for the four combinations of the following modules: a) the Truthful (TRU) and Data-driven (DD) Error Models; b) the Obedient (OB) and Task-based (TB) DA Model. The dialogs have been stored using an XML log format and uploaded into a Web tool for annotation and evaluation using the same procedure as for real data.

An initial look at the Dialog Act distribution in real data in comparison to the data produced by the DA Models (Figure 3) shows that OB only uses a subset of the DAs actually employed by real users, and it is easy to notice that the distribution of DAs is much more realistic when TB is used. Furthermore, we evaluated $KL-$divergence of the DA distributions produced by the Obedient and the Task-based model with respect to the real data distribution, following [10]. While the OB model obtained a $KL-$divergence of 0.926, the TB model almost reached zero (0.067), indicating that dialogs simulated by the latter are more similar to real dialogs.

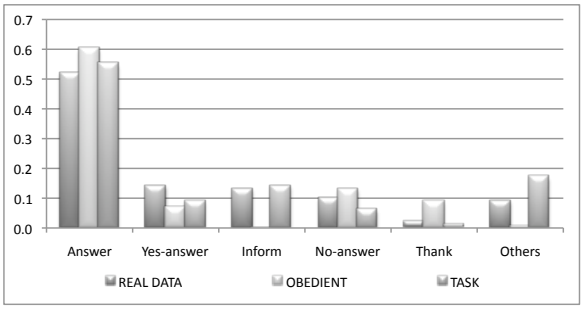

Figure 3: Dialog Act distribution in real dialogs vs dialogs simulated using different DA Models

Error robustness Concept-value precision P and recall R are reported in Table 2. While OB DA Model combined with TRU Error Model achieves an impractical 100% P and R, except in the case of Confirm as no value requires clarification due to the perfect communication channel, we also achieve unrealistic precision and recall in simulation regimes where a task-based model (TB) is combined with the truthful Error Model. Indeed, although the user's cooperativeness is made less than perfect by the presence of conversational fillers, clarification requests, or even the tendency to over-answer (TB+TRU column of Table 2), we still see very high P/R figures with respect to real data.

When comparing the Obedient and the Task-based DA Model against the Data-Driven Error Model, precision and recall appear much closer to those of real data in the latter case. In most cases however, concept-level accuracy of simulated dialogs appears overly "optimistic".

Task statistics Finally, task durations and completion rates

Table 2: Precision/Recall of selected concepts in different simulation models vs real dialogs. OB/TB = Obedient/Task-based DA Model; TRU/DD= Truthful/Data-Driven Error Model

	Real		OB+DD		TB+TRU		TB+DD	
Concept	P	R	P	R	P	R	P	R
Event	63.6	51.8	37.5	37.5	100	100	51.1	51.1
Location	64.2	71.5	71.4	71.4	100	100	71.2	71.2
day	55.0	59.9	85.8	85.8	85.4	94.1	65.3	73.5
month	67.5	66.5	78.9	78.9	78.3	78.3	57.2	62.8
Confirm	90.8	99.4	94.5	94.5	N/A	N/A	94.4	94.4

are compared in Table 3 against the same figures obtained for real data for the tasks supported by ADASearch: lodging enquiry/reservation and event enquiry. From here, we can clearly see that data-driven error simulation is vital to make dialogs realistic: it is sufficient to compare OB+DD and OB+TRU, and even more TB+DD and TB+TRU. This legitimizes the presence of an Error Model also in the case of a very sophisticated DA Model. For instance, clarification requests never appear within the TB Model when the truthful Error Model is applied: as perfect communication is assumed, concept confidence is highest. In addition, the number of turns required to complete tasks is much closer to real data when using the TB Dialog Act Model.

Table 3: Task duration/completion in simulated vs real dialogs

	Lodging Enquiry		Lodging Reserv		Event Enquiry		All
Model	#turns	TCR	#turns	TCR	#turns	TCR	TCR
OB+TRU	8.0±0.0	100	8.0±0.0	100	5.0±0.0	100	100
OB+DD	9.2±0.0	78.1	9.7±1.4	82.4	8.1±2.9	66.7	76.6
TB+TRU	8.6±0.8	94.3	9.2±0.9	92.5	5.4±0.5	89.5	92.5
TB+DD	**12.3±3.2**	**66.7**	**13.0±4.2**	**80.3**	**8.1±3.6**	58.8	**72.3**
Real data	**11.1±3.0**	**71.4**	**12.7±4.7**	**69.6**	**9.3±4.0**	**85.0**	**73.4**

5. Related Work

Initial work on interaction simulations for task-oriented dialog saw the introduction of bigram action models and their variations [5, 11, 12]; goal-directed simulation was later proposed by [6]. However, early simulators mainly focused on action (or dialog act) transition models and did not simulate ASR/SLU error; moreover, in such models probabilities tended to be handcrafted. Very recently, fully data-driven statistical simulators decoupling the state-transition model from noisy channel representations have been proposed [7, 13], however most studies choose the word level of abstraction; only [14] deals specifically with concepts but models fixed error rates for all concepts. To better control the training of Reinforcement Learning-based Dialog Managers, as in [2], we prefer to maintain the level of abstraction higher and focus on concept-level simulation.

The first comparative evaluation of User Simulators was achieved in [3], although without a focus on the impact of error simulation; we propose an evaluation where artificial and real dialogs are collected using the same Dialog Manager, and focus on the contribution of various combinations of Dialog Act and Error Models on dialog act distribution, concept accuracy, task completion rates and durations. However, as highlighted in [15], current evaluation metrics might not be sufficiently powerful to capture fine-grained simulator properties such as realism.

6. Conclusion

We design data-driven statistical dialog simulators for training statistical dialog managers from real user interactions. Our simulators support a modular combination of user-specific features with different models of dialog act and concept-value estimation as well as ASR/SLU error simulation.

We evaluate different combinations of user action generation and error simulation against real data obtained with a baseline dialog manager, obtaining realistic simulations in terms of dialog act distribution, concept accuracy and task completion rate. In particular, the dialog act distribution produced by our task-based dialog act model achieves a very low KL-divergence (0.067) with respect to real data; moreover, the task completion rate of 72.3 obtained by combining a task-based dialog act model with a data-driven error simulator is very close to that of real conversations in our domain (73.4). In future work, we will refine our simulators by studying additional features in the User Model designing and finer-grained Dialog Act models.

7. References

[1] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies," *Knowl. Eng. Rev.*, vol. 21, no. 2, 2006.

[2] S. Varges, S. Quarteroni, G. Riccardi, A. V. Ivanov, and P. Roberti, "Leveraging POMDPs trained with user simulations and rule-based dialogue management in a spoken dialogue system," in *Proc. SIGDIAL*, 2009.

[3] J. Schatzmann, K. Georgila, and S. Young, "Quantitative evaluation of user simulation techniques for spoken dialogue systems," in *Proc. SIGDIAL*, 2005.

[4] S. Quarteroni, G. Riccardi, S. Varges, and A. Bisazza, "An open-domain dialog act taxonomy," University of Trento, Tech. Rep. DISI-08-032, 2008.

[5] W. Eckert, E. Levin, and R. Pieraccini, "User modeling for spoken dialogue system evaluation," in *Proc. IEEE ASRU*, 1997.

[6] O. Pietquin, "A Framework for Unsupervised Learning of Dialogue Strategies," Ph.D. dissertation, Faculté Polytechnique de Mons, TCTS Lab (Belgique), 2004.

[7] J. Schatzmann, B. Thomson, and S. Young, "Error simulation for training statistical dialogue systems," in *Proc. ASRU*, 2007.

[8] S. Larsson and D. Traum, "Information State and dialogue management in the TRINDI Dialogue Move Engine Toolkit," *Nat. Lang. Eng.*, vol. 6, no. 3–4, 2000.

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

[10] H. Cuayahuitl, S. Renals, O. Lemon, and H. Shimodaira, "Human-computer dialogue simulation using hidden markov models," in *Proc. ASRU*, 2005.

[11] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, 2000.

[12] K. Scheffler and S. Young, "Corpus-based dialogue simulation for automatic strategy learning and evaluation," 2001.

[13] 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 & Language*, vol. 23, no. 4, 2009.

[14] D. Griol, Z. Callejas, and R. López-Cózar, "A comparison between dialog corpora acquired with real and simulated users," in *Proc. SIGDIAL*, 2009.

[15] H. Ai and D. Litman, "Comparing real-real, simulated-simulated, and simulated-real spoken dialogue corpora," in *In Proc. AAAI Workshop on Statistical and Empirical Approaches for SDS*, 2006.