An Unsupervised Approach to User Simulation: toward Self-Improving Dialog Systems

Sungjin Lee[1,2] and **Maxine Eskenazi**[1]
[1]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania
[2]Computer Science and Engineering, Pohang University of Science and Technology, South Korea
{sungjin.lee, max}@cs.cmu.edu[1], junion@postech.ac.kr[2]

Abstract

This paper proposes an unsupervised approach to user simulation in order to automatically furnish updates and assessments of a deployed spoken dialog system. The proposed method adopts a dynamic Bayesian network to infer the unobservable true user action from which the parameters of other components are naturally derived. To verify the quality of the simulation, the proposed method was applied to the Let's Go domain (Raux et al., 2005) and a set of measures was used to analyze the simulated data at several levels. The results showed a very close correspondence between the real and simulated data, implying that it is possible to create a realistic user simulator that does not necessitate human intervention.

1 Introduction

For the past decade statistical approaches to dialog modeling have shown positive results for optimizing a dialog strategy with real data by applying well-understood machine learning methods such as reinforcement learning (Henderson et al., 2008; Thomson and Young, 2010; Williams and Young, 2007b). User simulation is becoming an essential component in developing and evaluating such systems. In this paper we describe an unsupervised process to automatically develop user simulators. The motivation for this comes from the fact that many systems are presently moving from being simple lab simulations to actual deployed systems with real users. These systems furnish a constant flow of new data that needs to be processed in some way. Our goal is to minimize human intervention in processing this data. Previously, data had to be hand-annotated, a slow and costly process. Recently crowdsourcing has made annotation faster and less expensive, but all of the data still has to be processed and time must be spent in creating the annotation interface and tasks, and in quality control. Our goal is to process the metadata (e.g. user actions, goals, error typology) in an unsupervised manner. And our method eliminates the need for human transcription and annotation by inferring the user goal from grounding information. We also consider user actions as latent variables which are inferred based on observations from Automatic Speech Recognition (ASR). We used the above inferred user actions paired with the observed actions to build an error model. Since the focus of this work is placed on improving and evaluating the dialog strategy, error simulation can be carried out at the semantic level. This eliminates the need for transcription, which would have necessitated an error simulation at the surface level. The end result here will be a system that has as little human intervention as possible.

This paper is structured as follows. Section 2 describes previous research and the novelty of our approach. Section 3 elaborates on our proposed unsupervised approach to user simulation. Section 4 explains the experimental setup. Section 5 presents and discusses the results. Finally, Section 6 concludes with a brief summary and suggestions for future research.

2 Related Work

Previous user simulation studies can be roughly categorized into rule-based methods (Chung, 2005;

Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), pages 50–59,
Seoul, South Korea, 5-6 July 2012.

Lopez-Cozar et al., 2006; Schatzmann et al., 2007a) and data-driven methods (Cuayahuitl et al., 2005; Eckert et al., 1997; Jung et al., 2009; Levin et al., 2000; Georgila et al., 2006; Pietquin, 2004). Rule-based methods generally allow for more control over their designs for the target domain while data-driven methods afford more portability from one domain to another and are attractive for modeling user behavior based on real data. Although development costs for data-driven methods are typically lower than those of rule-based methods, previous data-driven approaches have still required a certain amount of human effort. Most intention-level models take a semantically annotated corpus to produce user intention without introducing errors (Cuayahuitl et al., 2005; Jung et al., 2009). Surface-level approaches need transcribed data to train their surface form and error generating models (Jung et al., 2009; Schatzmann et al., 2007b). A few studies have attempted to directly simulate the intention, surface, and error by applying their statistical methods on the recognized data rather than on the transcribed data (Georgila et al., 2006; Schatzmann et al., 2005). Although such approaches can avoid human intervention, the sole incorporation of erroneous user action can propagate those errors to the higher-level discourse features which are computed from them, and thus could result in less realistic user behavior. In this work, the true user action is treated as a hidden variable and, further, its associated dialog history is also viewed as latent so that the uncertainty of the true user action is properly controlled in a principled manner. Syed and Williams (2008) adopted the *Expectation Maximization* algorithm for parameter learning for a latent variable model. But their method still requires a small amount of transcribed data to learn the observation confusability, and it suffers from overfitting as a general property of maximum likelihood. To address this problem, we propose a Bayesian learning method, which requires no transcribed data.

3 Unsupervised Approach to User Simulation

Before describing each component in detail, we present the overall process of user simulation with an example in the Let's Go domain in Figure 1. To begin a dialog, the user simulator first sets the user goal by sampling the goal model. Then the user simulator engages in a conversation with the dialog system until the termination model ends it. At each turn, the termination model randomly determines whether the dialog will continue or not. If the dialog continues, the user model generates user actions at the predicate level with respect to the given user goal and system action. Having the user actions, the error template model transforms some user actions into other actions if necessary and determines which action will receive an incorrect value. After that, the error value model substantiates the values by drawing a confusable value if specified to be incorrect or by using the goal value. Finally, a confidence score will be attached to the user action by sampling the confidence score model which conditions on the correctness of the final user action.

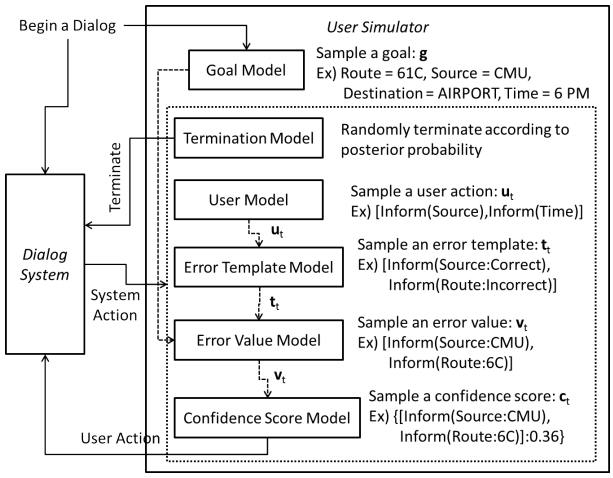

Figure 1: The overall process of user simulation in the Let's Go domain, where users call the spoken dialog system to get bus schedule information for Pittsburgh

3.1 Goal Model

The goal model is the first component to be defined in terms of the working flow of the user simulator. In order to generate a plausible user goal in accordance with the frequency at which it appears in a real situation, the dialog logs are parsed to look for the grounding information[1] that the users have provided. Since the representation of a user goal in this study is a vector of constraints required by a user, for example *[Route:61C, Source:CMU,*

[1] Specifically, we used explicitly confirmed information by the system for this study

Destination:AIRPORT, Time:6 PM], each time we encounter grounding information that includes the constraints used in the backend queries, this is added to the user goal. If two actions contradict each other, the later action overwrites the earlier one. Once all of the user goals in the data have been gathered, a discrete distribution over the user goal is learned using a maximum likelihood estimation. Because many variables later in this paper are discrete, a general notation of a conditional discrete distribution is expressed as follows:

$$p(\mathbf{x}_i|\mathbf{x}_{pa(i)}, \boldsymbol{\theta}) = \prod_{\mathbf{k},k'} \theta_{\mathbf{k},k'}^{\delta(pa(i),\mathbf{k})\delta(\mathbf{x}_i,k')} \quad (1)$$

where $\mathbf{k}$ represents the joint configuration of all the parents of i and $\delta(\cdot,\cdot)$ denotes *Kronecker* delta. Note that $\sum_{k'} \theta_{\mathbf{k},k'} = 1$. Given this notation, the goal model Λ can be written in the following form:

$$g \sim p(\mathbf{g}|\boldsymbol{\Lambda}) = \prod_k \lambda_k^{\delta(\mathbf{g},k)} \quad (2)$$

3.2 User Model

Having generated a user goal, the next task is to infer an appropriate user action for the given goal and system action. This is what the user model does. Since one of key properties of our unsupervised approach is that the true user actions are not observable, the user model should maintain a belief over the dialog state by taking into consideration the observed user actions. Inspired by (Williams et al., 2005), to keep the complexity of the user model tractable, a dynamic Bayesian network is adopted with several conditional independence assumptions, giving rise to the graphical structure which is shown in Figure 2. Unlike belief tracking in a dialog system, the user goal in a user simulation is pre-determined before the beginning of the dialog. As with most previous studies, this property allows the user model to deal with a predicate-level action consisting of a speech act and a concept (e.g. *[Inform(Source), Inform(Time)]*) and is only concerned about whether a given field is specified or not in the user goal (e.g. *Bus:Unspecified, Source:Specified*). This abstract-level handling enables the user model to employ exact inference algorithms such as the *junction tree* algorithm (Lauritzen and Spiegelhalter, 1988) for more efficient reasoning over the graphical structure.

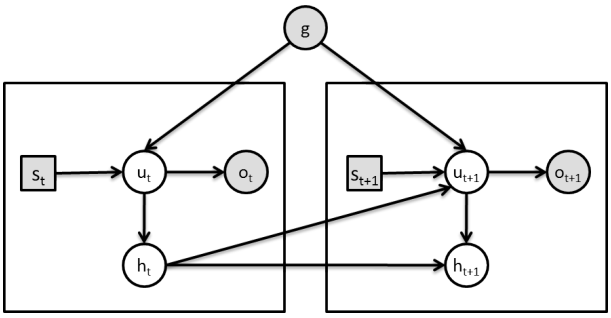

Figure 2: The graphical structure of the dynamic Bayesian network for the user model. $\mathbf{g}$ denotes the user goal and $\mathbf{s}_t, \mathbf{u}_t, \mathbf{h}_t, \mathbf{o}_t$ represents the system action, the user action, the dialog history, and the observed user action for each time slice, respectively. The shaded items are observable and the transparent ones are latent.

The joint distribution for this model is given by

$$\begin{aligned} &p(\mathbf{g}, \mathbf{S}, \mathbf{H}, \mathbf{U}, \mathbf{O}|\boldsymbol{\Theta}) \\ &\quad = p(\mathbf{h}_0|\boldsymbol{\pi}) \prod_t p(\mathbf{u}_t|\mathbf{g}, \mathbf{s}_t, \mathbf{h}_{t-1}, \boldsymbol{\phi}) \\ &\quad \cdot p(\mathbf{h}_t|\mathbf{h}_{t-1}, \mathbf{u}_t, \boldsymbol{\eta}) p(\mathbf{o}_t|\mathbf{u}_t, \boldsymbol{\zeta}) \end{aligned} \quad (3)$$

where a capital letter stands for the set of corresponding random variables, e.g., $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$, and $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\zeta}\}$ denotes the set of parameters governing the model[2].

For a given user goal, the user model basically performs an inference to obtain a marginal distribution over $\mathbf{u}_t$ for each time step from which it can sample the probability of a user action in a given context:

$$u_t \sim p(\mathbf{u}_t|\mathbf{g}, \mathbf{s}_1^t, \mathbf{u}_1^{t-1}, \boldsymbol{\Theta}) \quad (4)$$

where $\mathbf{s}_1^t$ denotes the set of system actions from time 1 to time t and $\mathbf{u}_1^{t-1}$ is the set of previously sampled user actions from time 1 to time $t-1$.

3.2.1 Parameter Estimation

As far as parameters are concerned, $\boldsymbol{\zeta}$ is a deterministic function that yields a fraction of an observed confidence score in accordance with the degree of agreement between $\mathbf{u}_t$ and $\mathbf{o}_t$:

$$p(\mathbf{o}_t|\mathbf{u}_t) = CS(\mathbf{o}_t) \cdot \left(\frac{|\mathbf{o}_t \cap \mathbf{u}_t|}{|\mathbf{o}_t \cup \mathbf{u}_t|} \right)^p + \epsilon \quad (5)$$

[2]Here, uniform prior distributions are assigned on $\mathbf{g}$ and $\mathbf{S}$

where $CS(\cdot)$ returns the confidence score of the associated observation and p is a control variable over the strength of disagreement penalty[3]. In addition, $\boldsymbol{\pi}$ and $\boldsymbol{\eta}$ are deterministically set by simple discourse rules, for example:

$$p(\mathbf{h}_t = \textit{Informed} | \mathbf{h}_{t-1}, \mathbf{u}_t) = \begin{cases} 1 & \text{if } \mathbf{h}_{t-1} = \textit{Informed} \text{ or } \mathbf{u}_t = \textit{Inform}(\cdot), \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

The only parameter that needs to be learned in the user model, therefore, is $\boldsymbol{\phi}$ and it can be estimated by maximizing the likelihood function (Equation 7). The likelihood function is obtained from the joint distribution (Equation 3) by marginalizing over the latent variables.

$$p(\mathbf{g}, \mathbf{S}, \mathbf{O} | \boldsymbol{\Theta}) = \sum_{\mathbf{H},\mathbf{U}} p(\mathbf{g}, \mathbf{S}, \mathbf{H}, \mathbf{U}, \mathbf{O} | \boldsymbol{\Theta}) \quad (7)$$

Since direct maximization of the likelihood function will lead to complex expressions with no closed-form solutions due to the latent variables, the *Expectation-Maximization* (EM) algorithm is an efficient framework for finding maximum likelihood estimates.

As it is well acknowledged, however, that overfitting can arise as a general property of maximum likelihood, especially when only a small amount of data is available, a *Bayesian* approach needs to be adopted. In a Bayesian model, any unknown parameter is given a prior distribution and is absorbed into the set of latent variables, thus it is infeasible to directly evaluate the posterior distribution of the latent variables and the expectations with respect to this distribution. Therefore a deterministic approximation, called *mean field* theory (Parisi, 1988), is applied.

In *mean field* theory, the family of posterior distributions of the latent variables is assumed to be partitioned into disjoint groups:

$$q(\mathbf{Z}) = \prod_{i=1}^{M} q_i(\mathbf{Z}_i) \quad (8)$$

where $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ denotes all latent variables including parameters and $\mathbf{Z}_i$ is a disjoint group. Amongst all distributions $q(\mathbf{Z})$ having the form of Equation 8, we then seek the member of this family for which the divergence from the true posterior distribution is minimized. To achieve this, the following optimization with respect to each of the $q_i(\mathbf{Z}_i)$ factors is to be performed in turn (Bishop, 2006):

$$\ln q_j^*(\mathbf{Z}_j) = E_{i \neq j}\big[\ln p(\mathbf{X}, \mathbf{Z})\big] + const \quad (9)$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denotes all observed variables and $E_{i \neq j}$ means an expectation with respect to the q distributions over all groups $\mathbf{Z}_i$ for $i \neq j$.

Now, we apply the *mean field* theory to the user model. Before doing so, we need to introduce the prior over the parameter $\boldsymbol{\phi}$ which is a product of *Dirichlet* distributions[4].

$$\begin{aligned} p(\boldsymbol{\phi}) &= \prod_{\mathbf{k}} Dir(\boldsymbol{\phi}_{\mathbf{k}} | \boldsymbol{\alpha}_{\mathbf{k}}^0) \\ &= \prod_{\mathbf{k}} C(\boldsymbol{\alpha}_{\mathbf{k}}^0) \prod_{l} \phi_{\mathbf{k},l}^{\alpha_{\mathbf{k}}^0 - 1} \end{aligned} \quad (10)$$

where $\mathbf{k}$ represents the joint configuration of all of the parents and $C(\boldsymbol{\alpha}_{\mathbf{k}}^0)$ is the normalization constant for the *Dirichlet* distribution. Note that for symmetry we have chosen the same parameter $\alpha_{\mathbf{k}}^0$ for each of the components.

Next we approximate the posterior distribution, $q(\mathbf{H}, \mathbf{U}, \boldsymbol{\phi})$ using a factorized form, $q(\mathbf{H}, \mathbf{U}) q(\boldsymbol{\phi})$. Then we first apply Equation 9 to find an expression for the optimal factor $q^*(\boldsymbol{\phi})$:

$$\begin{aligned} \ln q^*(\boldsymbol{\phi}) &= E_{\mathbf{H},\mathbf{U}}\big[\ln p(\mathbf{g}, \mathbf{S}, \mathbf{H}, \mathbf{U}, \mathbf{O}, \boldsymbol{\Theta})\big] + const \\ &= E_{\mathbf{H},\mathbf{U}}\left[\sum_t \ln p(\mathbf{u}_t | \mathbf{g}, \mathbf{s}_t, \mathbf{h}_{t-1}, \boldsymbol{\phi})\right] \\ &\quad + \ln p(\boldsymbol{\phi}) + const \\ &= \sum_t \sum_{i,j,k,l} \left(E_{\mathbf{H},\mathbf{U}}\big[\boldsymbol{\delta}_{i,j,k,l}\big] \ln \phi_{i,j,k,l} \right) \\ &\quad + \sum_{i,j,k,l} (\alpha_{i,j,k,l}^o - 1) \ln \phi_{i,j,k,l} + const \\ &= \sum_{i,j,k,l} \Bigg(\Big(E_{\mathbf{H},\mathbf{U}}[n_{i,j,k,l}] + (\alpha_{i,j,k,l}^o - 1) \Big) \\ &\quad \cdot \ln \phi_{i,j,k,l} \Bigg) + const \end{aligned} \quad (11)$$

[3] For this study, p was set to 1.0

[4] Note that priors over parameters for deterministic distributions (e.i., $\boldsymbol{\pi}$, $\boldsymbol{\eta}$,and $\boldsymbol{\zeta}$) are not necessary.

where $\boldsymbol{\delta}_{i,j,k,l}$ denotes $\delta(\mathbf{g},i)\delta(\mathbf{s}_t,j)\delta(\mathbf{h}_{t-1},k)$ $\delta(\mathbf{u}_t,l)$ and $n_{i,j,k,l}$ is the number of times where $\mathbf{g}=i,\mathbf{s}_t=j,\mathbf{h}_{t-1}=k$, and $\mathbf{u}_t=l$. This leads to a product of *Dirichlet* distributions by taking the exponential of both sides of the equation:

$$q^*(\boldsymbol{\phi}) = \prod_{i,j,k} Dir(\boldsymbol{\phi}_{i,j,k}|\boldsymbol{\alpha}_{i,j,k}), \qquad \alpha_{i,j,k,l} = \alpha^0_{i,j,k,l} + E_{\mathbf{H},\mathbf{U}}[n_{i,j,k,l}] \tag{12}$$

To evaluate the quantity $E_{\mathbf{H},\mathbf{U}}[n_{i,j,k,l}]$, Equation 9 needs to be applied once again to obtain an optimal approximation of the posterior distribution $q^*(\mathbf{H},\mathbf{U})$.

$$\begin{aligned}
\ln q^*(\mathbf{H},\mathbf{U}) &= E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{g},\mathbf{S},\mathbf{H},\mathbf{U},\mathbf{O},\boldsymbol{\Theta})\big] + const \\
&= E_{\boldsymbol{\phi}}\Big[\sum_t \ln p(\mathbf{u}_t|\mathbf{g},\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi}) \\
&\quad + \ln p(\mathbf{h}_t|\mathbf{h}_{t-1},\mathbf{u}_t) \\
&\quad + \ln p(\mathbf{o}_t|\mathbf{u}_t)\Big] + const \\
&= \sum_t \Big(E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{u}_t|\mathbf{g},\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\big] \\
&\quad + \ln p(\mathbf{h}_t|\mathbf{h}_{t-1},\mathbf{u}_t) \\
&\quad + \ln p(\mathbf{o}_t|\mathbf{u}_t)\Big) + const
\end{aligned} \tag{13}$$

where $E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{u}_t|\mathbf{g},\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\big]$ can be obtained using Equation 12 and properties of the *Dirichlet* distribution:

$$\begin{aligned}
&E_{\boldsymbol{\phi}}\big[\ln p(\mathbf{u}_t|\mathbf{g},\mathbf{s}_t,\mathbf{h}_{t-1},\boldsymbol{\phi})\big] \\
&\quad = \sum_{i,j,k,l} \boldsymbol{\delta}_{i,j,k,l} E_{\boldsymbol{\phi}}\big[\ln \phi_{i,j,k,l}\big] \\
&\quad = \sum_{i,j,k,l} \boldsymbol{\delta}_{i,j,k,l} (\psi(\alpha_{i,j,k,l}) - \psi(\hat{\boldsymbol{\alpha}}_{i,j,k}))
\end{aligned} \tag{14}$$

where $\psi(\cdot)$ is the digamma function with $\hat{\boldsymbol{\alpha}}_{i,j,k} = \sum_l \alpha_{i,j,k,l}$. Because computing $E_{\mathbf{H},\mathbf{U}}[n_{i,j,k,l}]$ is equivalent to summing each of the marginal posterior probabilities $q^*(\mathbf{h}_{t-1},\mathbf{u}_t)$ with the same configuration of conditioning variables, this can be done efficiently by using the *junction tree* algorithm. Note that the expression on the right-hand side for both $q^*(\boldsymbol{\phi})$ and $q^*(\mathbf{H},\mathbf{U})$ depends on expectations computed with respect to the other factors. We will therefore seek a consistent solution by cycling through the factors and replacing each in turn with a revised estimate.

3.3 Error Model

The purpose of the error model is to alter the user action to reflect the prevalent speech recognition and understanding errors. The error generation process consists of three steps: the error model first generates an error template then fills it with erroneous values, and finally attaches a confidence score.

Given a user action, the error model maps it into a distorted form according to the probability distribution of the error template model $\boldsymbol{\Omega}$:

$$\mathcal{T}(u) \sim p(\mathcal{T}(u)|u) = \prod_{k,k'} \omega_{k,k'}^{\delta(u,k)\delta(\mathcal{T}(u),k')} \tag{15}$$

where $\mathcal{T}(\cdot)$ is a random function that maps a predicate of the user action to an error template, e.g. $\mathcal{T}$(*Inform(Time)*) $\rightarrow$ *Inform(Route:incorrect)*. To learn the parameters, the hidden variable $\mathbf{u}_t$ is sampled using Equation 4 for each observation $\mathbf{o}_t$ in the training data and the value part of each observation is replaced with a binary value representing its correctness with respect to the user goal. This results in a set of complete data on which the maximum likelihood estimates of $\boldsymbol{\Omega}$ are learned.

With the error template provided, next, the error model fills it with incorrect values if necessary following the distribution of the error value model $\boldsymbol{\Lambda}$ which is separately defined for each concept, otherwise it will keep the correct value:

$$\mathcal{C}(v) \sim p(\mathcal{C}(v)|v) = \prod_{k,k'} \lambda^{\delta(v,k)\delta(\mathcal{C}(v),k')} \tag{16}$$

where $\mathcal{C}(\cdot)$ is a random function which maps a correct value to a confusable value, e.g. $\mathcal{C}$(*Forbes*) $\rightarrow$ *Forward*. As with the error template model, the parameters of the error value model are also easily trained on the dataset of all pairs of a user goal value and the associated observed value. Because no error values can be observed for a given goal value, an unconditional probability distribution is also trained as a backoff.

Finally, the error model assigns a confidence score by sampling the confidence score model $\boldsymbol{\Gamma}$

which is separately defined for each concept:

$$s \sim p(s|c) = \prod_{k,k'} \gamma^{\delta(c,k)\delta(s,k')} \quad (17)$$

where s denotes the confidence score and c represents the correctness of the value of the user action which is previously determined by the error template model. Since two decimal places are used to describe the confidence score, the confidence score model is represented with a discrete distribution. This lends itself to trivial parameter learning similar to other models by computing maximum likelihood estimates on the set of observed confidence scores conditioned on the correctness of the relevant values.

In sum, for example, having a user action *[Inform(Source:Forbes), Inform(Time:6 PM)]* go through the sequence of aforementioned models possibly leads to *[Inform(Source:Forward), Inform(Route:6C)]*.

3.4 Termination Model

Few studies have been conducted to estimate the probability that a dialog will terminate at a certain turn in the user simulation. Most existing work attempts to treat a termination initiated by a user as one of the dialog actions in their user models. These models usually have a limited dialog history that they can use to determine the next user action. This *Markov* assumption is well-suited to ordinary dialog actions, each generally showing a correspondence with previous dialog actions. It is not difficult, however, to see that more global contexts (e.g., cumulative number of incorrect confirmations) will help lead a user to terminate a failed dialog. In addition, the termination action occurs only once at the end of a dialog unlike the other actions. Thus, we do not need to put the termination action into the user model. In order to easily incorporate many global features involving an entire dialog (Table 1) into the termination model, the *logistic regression* model is adapted. At every turn, before getting into the user model, we randomly determine whether a dialog will stop according to the posterior probability of the termination model given the current dialog context.

Feature	Description
NT	Number of turns
RIC	Ratio of incorrect confirmations
RICW	Ratio of incorrect confirmations within a window
RNONU	Ratio of non-understanding
RNONUW	Ratio of non-understanding within a window
ACS	Averaged confidence score
ACSW	Averaged confidence score within a window
RCOP	Ratio of cooperative turns
RCOPW	Ratio of cooperative turns within a window
RRT_C	Ratio of relevant system turns for each concept
RRTW_C	Ratio of relevant system turns for each concept within a window
NV_C	Number of values appeared for each concept

Table 1: A description of features used for a logistic regression model to capture the termination probability. The window size was set to 5 for this study.

4 Experimental Setup

4.1 Data

To verify the proposed method, three months of data from the Let's Go domain were split into two months of training data and one month of test data. Also, to take the error level into consideration, we classified the data into four groups according to the averaged confidence score and used each group of data to build a different error model for each error level. For comparison purposes, simulated data was generated for both training and test data by feeding the same context of each piece of data to the proposed method. Due to the characteristics of the bus schedule information domain, there are a number of cases where no bus schedule is available, such as requests for uncovered routes and places. Such cases were excluded for clearer interpretation of the result, giving us the data sets described in Table 2.

4.2 Measures

To date, a variety of evaluation methods have been proposed in the literature (Cuayahuitl et al., 2005; Jung et al., 2009; Georgila et al., 2006; Pietquin and

	Training data	Test data
Number of dialogs	1,275	669
Number of turns	9,645	5,103

Table 2: A description of experimental data sets.

	Training data		Test data	
Error Mark	w/o	w/	w/o	w/
EP	58.13	45.12	54.44	41.86
ER	58.40	45.33	54.61	41.99
F-Score	58.27	45.22	54.52	41.93

Table 3: Expected precision, expected recall and F-Score

Hastie, 2011; Schatzmann et al., 2005; Williams, 2007a). Nevertheless, it remains difficult to find a suitable set of evaluation measures to assess the quality of the user simulation. We have chosen to adopt a set of the most commonly used measures. Firstly, expected precision (EP), expected recall (ER) and F-Score offer a reliable method for comparing real and simulated data even though it is not possible to specify the levels that need to be satisfied to conclude that the simulation is realistic. These are computed by comparison of the simulated and real user action for each turn in the corpus:

$$EP = 100 * \frac{\textit{Number of identical actions}}{\textit{Number of simulated actions}} \quad (18)$$

$$ER = 100 * \frac{\textit{Number of identical actions}}{\textit{Number of real actions}} \quad (19)$$

$$\textit{F-Score} = 100 * \frac{2 * EP * ER}{EP + ER} \quad (20)$$

Next, several descriptive statistics are employed to show the closeness of the real and simulated data in a statistical sense. The distribution of different user action types, turn length and confidence score can show constitutional similarity. It is still possible, however, to be greatly different in their interdependence and cause quite different behavior at the dialog level even though there is a constitutional similarity. Therefore, the dialog-level statistics such as dialog completion rate and averaged dialog length were also computed by running the user simulator with the Let's Go dialog system.

5 Results

As mentioned in Section 4.2, expected precision and recall were measured. Whereas previous studies only reported the scores computed in the predicate level, i.e. speech act and concept, we also measured the scores based on the output of the error template model which is the predicate-level action with an indicator of the correctness of the associated value (Figure 1). The result (Table 3) shows a moderate balance between agreement and variation which is a very desirable characteristic of a user simulator since a simulated user is expected not only to resemble real data but also to cover diverse unseen behavior to a reasonable extent. As a natural consequence of the increased degree of freedom, the scores considering error marking are consistently lower. In addition, the results of test data are slightly lower than those of training data, as expected, yet a suitable balance remains.

Next, the comparative distributions of different actions between real and simulated data are presented for both training and test data (Figure 3). The results are also based on the output of the error template model to further show how errors are distributed over different actions. The distributions of simulated data either from training or test data show a close match to the corresponding real distributions. Interestingly, even though the error ratio of the test data is noticeably different from that of the training data, the proposed method is still able to generate similar results. This means the variables and their conditional probabilities of the proposed method were designed and estimated properly enough to capture the tendency of user behavior with respect to various dialog contexts. Moreover, the comparison of the turn length distribution (Figure 4) indicates that the simulated data successfully replicated the real data for both training and test data. The results of confidence score simulation are presented in Figure 5[5]. For both training and test data, the simulated confidence score displays forms that are very similar to the real ones.

Finally, to confirm the resemblance on the dialog level, the comparative results of dialog completion rate and averaged dialog length are summarized in Table 4. As shown in the dialog completion result, the simulated user is a little harder than the real user

[5]Due to the space limitation, the detailed illustrations for each action type are put in Appendix A.

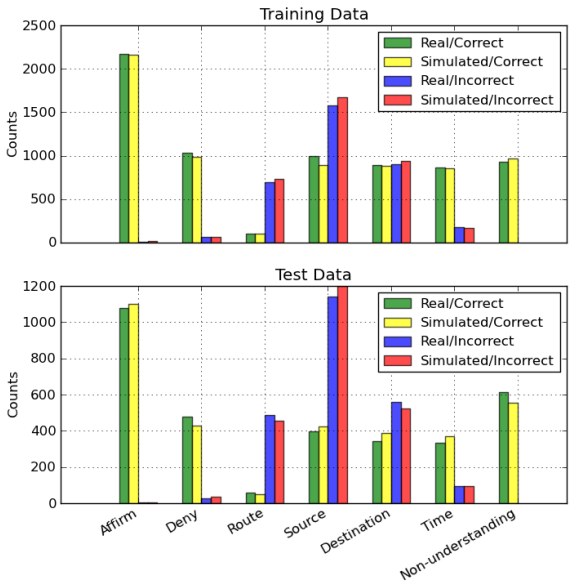

Figure 3: A comparison of the distribution of different actions between real and simulated data for both training and test data

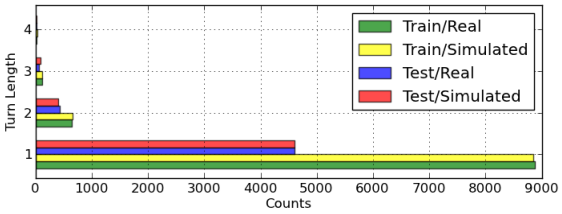

Figure 4: A comparison of the distribution of turn length between real and simulated data for both training and test data

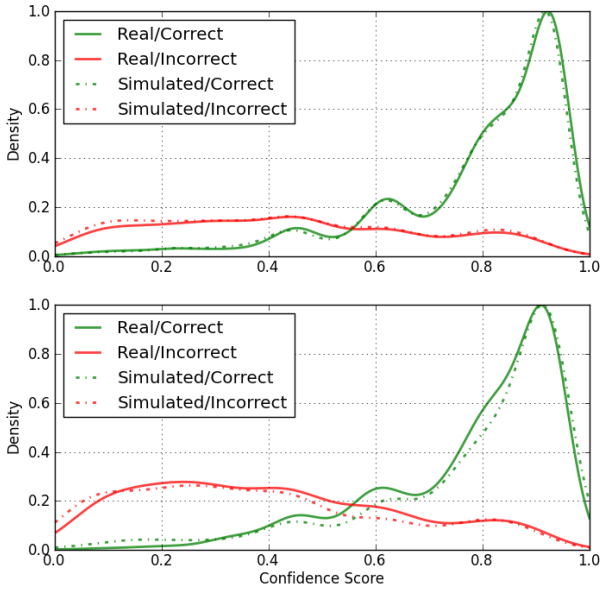

Figure 5: A comparison of the distribution of confidence score between real and simulated data for both training and test data

	Real		**Simulated**	
DCR (%)	59.68		55.04	
ADL	mean	std.	mean	std.
Success	10.62	4.59	11.08	5.10
Fail	7.75	6.20	7.75	8.64
Total	9.46	5.48	9.50	7.12

Table 4: A comparison of dialog completion rate (DCR) and averaged dialog length (ADL) which is presented according to the dialog result.

to accomplish the purpose. Also, the variation of the simulated data as far as turn length is concerned was greater than that of the real data, although the averaged lengths were similar to each other. This might indicate the need to improve the termination model. The proposed method for the termination model is confined to incorporating only semantic-level features but a variety of different features would, of course, cause the end of a dialog, e.g. system delay, acoustic features, spatial and temporal context, weather and user groups.

6 Conclusion

In this paper, we presented a novel unsupervised approach for user simulation which is especially desirable for real deployed systems. The proposed method can cover the whole pipeline of user simulation on the semantic level without human intervention. Also the quality of simulated data has been demonstrated to be similar to the real data over a number of commonly employed metrics. Although the proposed method does not deal with simulating N-best ASR results, the extension to support N-best results will be one of our future efforts, as soon as the Let's Go system uses N-best results. Our future work also includes evaluation on improving and evaluating dialog strategies. Furthermore, it would be scientifically more interesting to compare the proposed method with a supervised approach using a corpus with semantic transcriptions. On the other hand, as an interesting application, the proposed user model could be exploited as a part of belief tracking in a spoken dialog system since it also considers a user action to be hidden.

Acknowledgments

We would like to thank Alan Black for helpful comments and discussion. This work was supported by the second Brain Korea 21 project.

References

C. Bishop, 2006. *Pattern Recognition and Machine Learning*. Springer.

G. Chung, 2004. Developing a Flexible Spoken Dialog System Using Simulation. *In Proceedings of ACL*.

H. Cuayahuitl, S. Renals, O. Lemon, H. Shimodaira, 2005. Humancomputer dialogue simulation using hidden Markov models. *In Proceedings of ASRU*.

W. Eckert, E. Levin, R. Pieraccini, 1997. User modeling for spoken dialogue system evaluation. *In Proceedings of ASRU*.

K. Georgila, J. Henderson, O. Lemon, 2006. User simulation for spoken dialogue systems: Learning and evaluation. *In Proceedings of Interspeech*.

J. Henderson, O. Lemon, K. Georgila, 2008. Hybrid Reinforcement / Supervised Learning of Dialogue Policies from Fixed Datasets. *Computational Linguistics*, 34(4):487-511

S. Jung, C. Lee, K. Kim, M. Jeong, G. Lee, 2009. Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer Speech and Language*, 23(4):479–509.

S. Lauritzen and D. J. Spiegelhalter, 1988. Local Computation and Probabilities on Graphical Structures and their Applications to Expert Systems. *Journal of Royal Statistical Society*, 50(2):157–224.

E. Levin, R. Pieraccini, W. Eckert, 2000. A stochastic model of humanmachine interaction for learning dialogstrategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11-23.

R. Lopez-Cozar, Z. Callejas, and M. McTear, 2006. Testing the performance of spoken dialogue systems by means of an articially simulated user. *Articial Intelligence Review*, 26(4):291-323.

G. Parisi, 1988. *Statistical Field Theory*. Addison-Wesley.

O. Pietquin, 2004. A Framework for Unsupervised Learning of Dialogue Strategies. *Ph.D. thesis, Faculty of Engineering*.

O. Pietquin and H. Hastie, 2011. A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*.

A. Raux, B. Langner, D. Bohus, A. W Black, and M. Eskenazi, 2005. Let's Go Public! Taking a Spoken Dialog System to the Real World. *In Proceedings of Interspeech*.

J. Schatzmann, K. Georgila, S. Young, 2005. Quantitative evaluation of user simulation techniques for spoken dialogue systems. *In Proceedings of SIGdial*.

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

J. Schatzmann, B. Thomson, S. Young, 2007. Error simulation for training statistical dialogue systems. *In Proceedings of ASRU*.

U. Syed and J. Williams, 2008. Using automatically transcribed dialogs to learn user models in a spoken dialog system. *In Proceedings of ACL*.

B. Thomson and S. Young, 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562-588.

J. Williams, P. Poupart, and S. Young, 2005. Factored Partially Observable Markov Decision Processes for Dialogue Management. *In Proceedings of Knowledge and Reasoning in Practical Dialogue Systems*.

J. Williams, 2007. A Method for Evaluating and Comparing User Simulations: The Cramer-von Mises Divergence. *In Proceedings of ASRU*.

J. Williams and S. Young, 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393-422.

Appendices

Appendix A. Distribution of confidence score for each concept

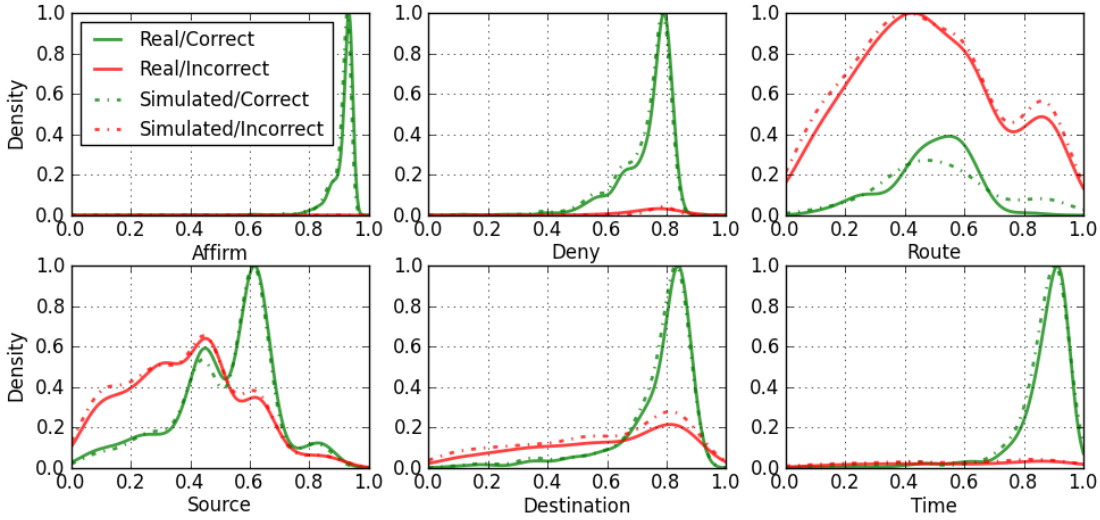

Figure 6: A comparison of the distribution of confidence score between real and simulated data for the training data

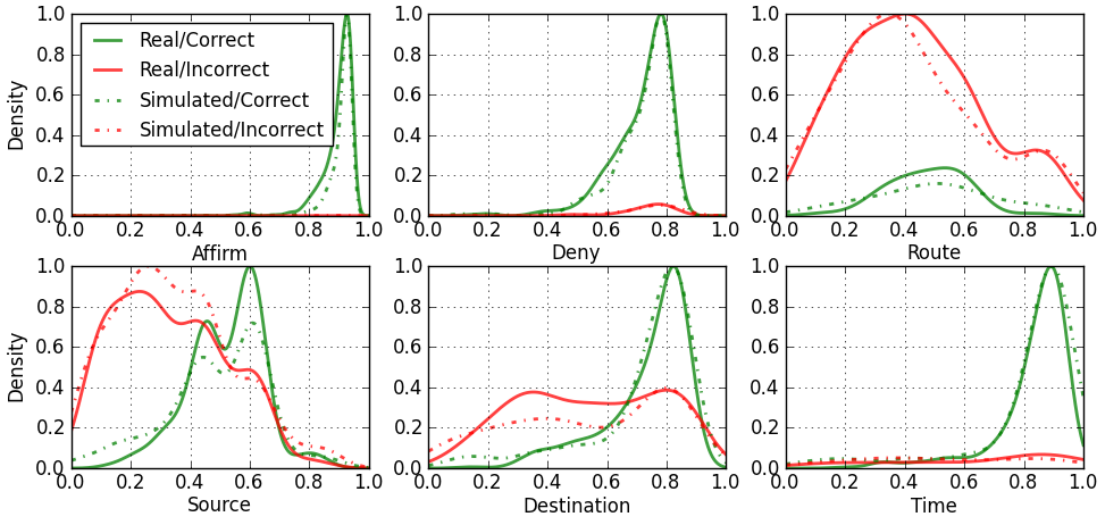

Figure 7: A comparison of the distribution of confidence score between real and simulated data for the test data