A User Simulator based on VoiceXML for evaluation of spoken dialog systems

Akinori Ito, Keisuke Shimada, Motoyuki Suzuki, Shozo Makino

Graduate School of Engineering
Tohoku University, Sendai, Japan
{aito,shike,moto,makino}@makino.ecei.tohoku.ac.jp

Abstract

This paper describes a user simulator based on analysis of VoiceXML description. A user simulator is a method to evaluate a spoken dialog system without the use of human evaluators. The new feature of our simulator is that it uses a VoiceXML description that describes the dialog system's behavior. By using the VoiceXML description, the proposed simulator can be used for any dialog system that works with VoiceXML. We constructed a prototype of the user simulator and carried out an evaluation experiment. The experimental result showed that the dialog between the simulator and the dialog system had similar properties to that between human subjects and the dialog system.

Index Terms: spoken dialog system, user simulator, VoiceXML, system evaluation

1. Introduction

Spoken dialog systems provide a user with an easy way to operate various equipment and electric appliances[1] or to access automated telephone-based services[2]. To develop a reliable spoken dialog system, it is indispensable to run extensive tests. Of the many items to be checked; stability of the system, average time of a session and recognition performance are but a few. As a spoken dialog system is designed to interact with a human, performing an extensive test of such a system requires many human subjects which, in turn, requires a lot of time and money.

Therefore, to reduce the amount of experiments involving human subjects, a simulator-based evaluation of spoken dialog systems has been proposed. López-Cózar et al. proposed a user simulator[3] to evaluate dialog system SAPLEN[4], which performs dialog pertaining to orders and queries in fast food restaurants.

The user simulator by López-Cózar et al. is based on three data: the scenario corpus, the utterance corpus and the action rules. The scenario corpus is a set of semantic frames, of which each frame contains a goal for the dialog. The user simulator performs a dialog with a spoken dialog system according to a scenario chosen from this corpus. Second, the utterance corpus is a set of recorded user utterances. The user generator makes an utterance by playing an utterance in this corpus. Third, the action rules determine the next action to be taken by the user simulator when a prompt is given by the spoken dialog system.

There are two problems relating to the user simulator proposed by López-Cózar et al. One problem is that it requires all user utterances to be recorded beforehand. In the developmental processes of a dialog system, the addition or alteration of words in the vocabulary often occurs. To accommodate this, a developer has to record the human voice of the additional words in order to evaluate the revised dialog system. The other problem is that the action rules are task-dependent. As such, the developer has to write the rules by hand when evaluating a dialog system of a different domain.

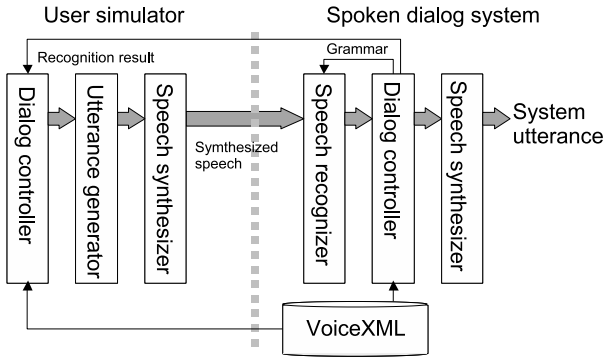

Figure 1: *Overview of the user simulator.*

In this paper, we propose a user simulator that can be used for evaluation of spoken dialog systems that uses VoiceXML[5] description. Our user simulator uses a speech synthesizer to generate the simulated user utterances. Moreover, the simulator exploits the VoiceXML description that describes the behavior of the spoken dialog systems to be evaluated.

2. Behavior of the proposed user simulator

2.1. Overview

Figure 1 shows an overview of the proposed user simulator. There are several assumptions on the spoken dialog system to be evaluated by the user simulator:

- The dialog system is based on VoiceXML, which means that all the behaviors of the system can be written in VoiceXML.
- The strategy of the dialog system is restricted to a system-initiative, item-by-item query.
- The dialog system always makes a confirmation whenever the dialog flow branches.
- The recognition results by the dialog system can be observed directly from an external program (by looking in the log file, for example) while performing a dialog.

The assumptions except the first one are too strong to use the proposed user simulator for all dialog systems. These points have to be improved in future versions of the simulator.

```
<form id="beverage_service">
  <field name="beverage">
    <prompt>
      What would you like to drink?
    </prompt>
    <grammar src="drink_grammar.gsl"/>
  </field>
  <field name="num">
    <prompt>How many cups?</prompt>
    <grammar src="num_grammar.gsl"/>
  </field>
  <filled>
    <goto next="#yesno">
  </filled>
</form>
```

Figure 2: An example of a dialog description in VoiceXML (voice input)

Now, let us explain the behavior of the user simulator. First, the simulator reads the VoiceXML file that describes the behavior of the spoken dialog system. Then, the simulator analyzes the description and gathers the following information:

- All forms and variables corresponding to the items to be filled
- All grammar used to recognize utterances

At this stage the simulator waits for the dialog system's prompt. When the dialog system utters the first prompt message, the simulator generates the goal of the dialog randomly using the variables and grammar of the dialog description written in VoiceXML. When the dialog system asks the user the value of an item, the simulator utters it according to the dialog goal. When the dialog system makes a confirmation utterance, the simulator compares the values of variables in the dialog system with the dialog goal. If the recognized items differ from the items in the dialog goal, then the simulator makes corrective utterances. Otherwise, the simulator utters 'yes' and the dialog terminates.

2.2. VoiceXML description

Figure 2 and 3 show examples of the form part of the VoiceXML file. These examples are a part of a task that involves accepting an order for a beverage. From the assumption, the dialog strategy is restricted to system-initiative item-by-item style. Therefore, we can assume that one field will correspond to one query.

Figure 2 is an example pertaining to items. In this example, the dialog system asks the user the preference of a beverage and quantity. The query of each item is specified by a `<field>` tag, and the queries are enclosed by a `<form>` tag. Within one `<field>` tag there is a `<prompt>` tag and a `<grammar>` tag. The `<prompt>` tag specifies the prompt message to be played by the speech synthesizer, and the `<grammar>` tag specifies the grammar to be matched to the user's utterance.

Figure 3 shows an example of a confirmation. In this example, the message in the `<prompt>` tag presents the user with the recognition results, and the dialog system waits for the user's confirmation utterance. When the confirmation is accepted, the dialog state branches (the `<if>` tag) according to the user's utterance. When the answer is 'yes', the dialog system exits and the values

```
<form id="yesno">
  <field name="confirm1">
    <prompt>
      I will bring <value expr="num">
      <value expr="beverage">. OK?
    </prompt>
    <grammar src="confirm_grammar.gsl"/>
  </field>
  <filled>
    <if cond="confirm1 == yes">
      <prompt>
        OK. I will bring <value expr="num">
        <value expr="beverage">.
      </prompt>
      <exit/>
    <else/>
      <goto next="#beverage_service"/>
    </if>
  </filled>
</form>
```

Figure 3: An example of a dialog description in VoiceXML (confirmation)

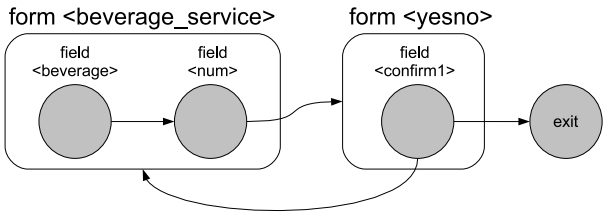

Figure 4: State transition of the sample VoiceXML

of the variables (in this example, `beverage` and `num`) are sent to the backend system.

The dialog flow written in VoiceXML can be regarded as a kind of finite state automata. The FSA correspond to the above example as shown in Figure 4. The `<field>` tags correspond to states in the FSA, and the `<form>` tags are groups of states. The state transition happens when the dialog system accepts a user's utterance.

2.3. Behavior of the user simulator

The user simulator holds the same FSA as the dialog system. The difference is that the simulator *makes an utterance* when moving from one state to another, while the dialog system *accepts an utterance*.

First, the simulator generates the goal of the dialog. Here, the goal of the dialog is defined as a set of field variables that appear in the VoiceXML description files along with the values of the variables. While López-Cózar's simulator uses a corpus of dialog goals, our simulator generates any combination of variables and their values, randomly. The available values of a variable are obtained from the grammar specified in the field.

The simulator behaves differently according to whether the current state is a confirmation state or not. If the current state is

not a confirmation state, then the simulator makes an utterance according to the field variable associated with the current state. To make an utterance, the simulator references the grammar of the current state, and chooses a sentence randomly that corresponds to the value of the field variable.

The simulator regards the current state to be a confirmation state if the name of the field contains a string 'confirm'. If the current state is a confirmation state, the simulator first checks the contents of the <prompt> tag. The prompt message must contain <value> tags to confirm the values of the recognized items. Therefore, the simulator looks up the <value> tags in the prompt message, and the variables in the <value> tags are regarded as the items to be confirmed. The simulator compares the values of the variables in the dialog system with the corresponding values in the dialog goal. If all the values are equal, the simulator simply utters 'yes.' Otherwise, it utters 'no' and moves to the input state according to the VoiceXML description.

2.4. What the simulator can and cannot do

As explained above, the simulator behaves according to the VoiceXML description for the spoken dialog system. That means that the simulator assumes the role of the user who completely complies with the scenario of the dialog system. It can be used to measure the word recognition rate, average turn, task completion rate etc., or to investigate the effect of a different speaker (by changing the speaker model of the speech synthesizer) and environmental noise. The simulator can also be used to test the reliability of the dialog system by performing multiple dialogs continuously.

Conversely, it is not capable of processing utterances that the grammar of the system does not cover, or to evaluate the behavior of the system against out-of-vocabulary utterances. To achieve these measures, we need to apply other user models than the VoiceXML description. This is an interesting and important issue to be solved in future work.

3. Experiments

3.1. Overview

We carried out experiments to evaluate the user simulator. The criterion of the evaluation is how the evaluation of the dialog system employing a simulator is similar to that of the system using human subjects.

First, we carried out an experiment of recognizing synthesized speech to confirm whether speech synthesis can be used instead of playing recorded speech. In this experiment, word accuracy of synthesized speech is compared to that of the human voice.

Next, we carried out dialog experiments involving five small tasks using both the user simulator and human subjects. Then, the number of user utterances and the task completion rate of the simulator are compared with those of the human subjects.

3.2. Recognition of synthesized speech

Table 1 shows the conditions of the experiment. We examined two speech synthesizers. GalateaTalk, which is based on HMM-based speech synthesis technology, is a part of spoken dialog system Galatea[8]. SmartVoice XP is a commercial voice recognition and synthesis engine produced by NEC. The task was recognition of continuous, read, Japanese speech (newspaper article).

The word accuracy for five synthesized voices and one human voice is shown in Figure 5. These results showed that the word accuracy for SmartVoice was not good, but that for GalateaTalk is as good as that for a natural human voice. As quality of the voice synthesized by SmartVoice is not less than that by GalateaTalk, this bias in the result might be caused by differences in the speech database from which the speech synthesizer is trained.

Table 1: Experimental conditions of the word recognition experiment

Evaluation sentences	105 sentences from Mainichi Shimbun newspaper[6]
Recognition engine	Julius 3.4-multipath[7]
Acoustic model	2000 states, 16 mixture PTM model for Japanese
Language model	20k vocab. trigram trained from 75 months of Mainichi Shimbun
Speech synthesizer	GalateaTalk (2 males, 1 female), SmartVoice XP (1 male, 1 female)
Human subject	1 male from JNAS database[6]

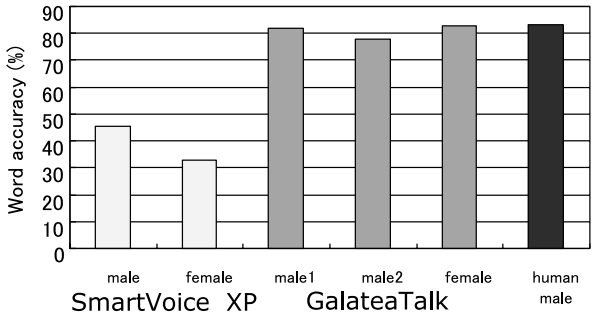

Figure 5: Recognition results for various synthesized voices

3.3. Dialog experiments

Next, we carried out machine-machine and human-machine dialog experiments. Table 2 shows the tasks used in the experiments. Tasks 1–4 are commands given to an intelligent care robot[9]. Tasks 5 and 6 are ticket reservations for the Tohoku Shinkansen Superexpress. The difference between task 5 and 6 is that task 5 asks only the destination and the seat (reserved or non-reserved) while in task 6 the origin of the train is also asked. When the seat is reserved, the system also asked the preference regarding being seated in a smoking car. The column '#items' shows the number of items to be asked, and N_m shows the minimum number of user utterances.

In this experiment we used GalateaTalk (male 1 and male 2 as the voices) as the speech synthesizer. The number of machine-machine dialogs was 165. We conducted the speech recognition test via generated speech files by the synthesizer (i.e. no environmental noise).

As for the human-machine dialog experiment, 8 subjects made 404 dialogs in total. The dialog experiment was carried out in a

Table 2: Tasks used in the dialog experiments

Task No.	task description	# items	N_m
1	Throw garbage away	1	2
2	Open/close curtain	2	3
3	Bring a cup or a chair	2	3
4	Bring something to drink	2	3
5a	Reservation of train 1 (non-reserved)	2	3
5b	Reservation of train 1 (reserved)	3	4
6a	Reservation of train 2 (non-reserved)	3	4
6b	Reservation of train 2 (reserved)	4	5

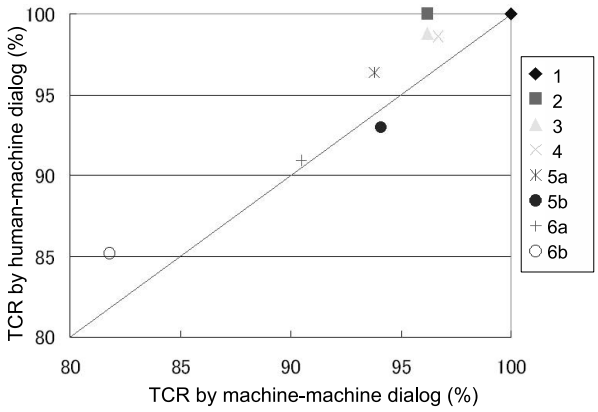

Figure 6: Comparison of task completion rate

silent room.

The comparison of task completion rate (TCR) is shown in Figure 6. From this result it is clear that the TCR for human-machine dialog and that of machine-machine dialog are strongly related. However, the absolute value of the TCR of human-machine dialog is slightly higher than that of machine-machine dialog. The decrease in TCR in machine-machine dialog was caused by misrecognition of confirmation utterances, where confirmation utterances were recognized as 'yes'.

Figure 7 shows the average number of user utterances for both dialogs. As the tasks used in this experiment were relatively simple, the results were similar to the N_m of each task. This result shows that the average number of user utterances can be precisely estimated by a machine-machine dialog experiment.

4. Conclusion

The user simulator based on VoiceXML was described. This simulator exploits the description of dialog written in VoiceXML, and behaves as a user of that dialog system. It utilizes a speech synthesizer to generate user utterances, which enabled us to evaluate a dialog system without recording any real voice. The recognition experiment proved that synthesized speech could be used as an alternative to a recorded human voice. In addition, we carried out human-machine and machine-machine dialog experiments. The experimental results showed that the result of the human-machine

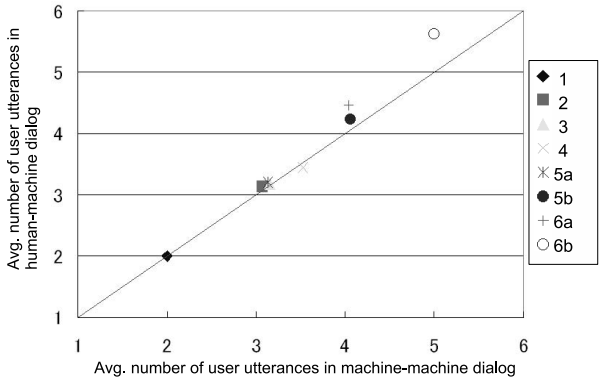

Figure 7: Comparison of number of user utterances

dialog could be estimated using the result of the machine-machine dialog experiment.

The remaining problems are that the simulator cannot perform out-of-task behavior or out-of-vocabulary utterances. To achieve a more 'real-human-like' simulator, we must model the user behavior in a different way.

5. References

[1] I. Potamitis, K. Georgila, N. Fakotakis and G. Kokkinakis, "An Integrated System for Smart-Home Control of Appliances Based on Remote Speech Interaction," *Proc. Eurospeech 2003, Geneva,* 2003.

[2] A. Raux, B. Langner, A. Black and M. Eskenazi, "LET'S GO: Improving Spoken Dialog Systems for the Elderly and Non-natives," *Proc. Eurospeech 2003, Geneva,* 2003.

[3] R. López-Cózar, A. de la Torre, J.C. Segura, A.J. Rubio, V. Sánchez, "Testing Dialogue Systems By Means of Automatic Generation of Conversations," *Interacting with Computers*, vol. 14, no. 5, pp. 521–546, 2002.

[4] R. López-Cózar, A. J. Rubio, P. García, J. C. Segura, "A New Word-Confidence Threshold Technique to Enhance the Performance of Spoken Dialogue Systems," *Proc. Eurospeech99*, pp. 1395–1398, 1999.

[5] VoiceXML forum. http://www.voicexml.org/

[6] K.Itou, et al., "The design of the newspaper-based Japanese large vocabulary continuous speech recognition corpus," *Proc. ICSLP98*, pp.3261–3264, 1998.

[7] A. Lee, T. Kawahara, and K. Shikano, "Julius — an open source real-time large vocabulary recognition engine," *Proc. Eurospeech2001*, pp. 1691-1694, 2001.

[8] S. Kawamoto, et al., "Open-source software for developing anthropomorphic spoken dialog agent," *Proc. PRICAI-02, International Workshop on Lifelike Animated Agents,* pp. 64–69, 2002.

[9] Y. Hiroi et al., "A Patient Care Service Robot System Based on a State Transition Architecture," *Proc. 2nd Int. Conf. Mechatronics and Information Technology,* pp. 231–236, 2003.