USER SIMULATION FOR THE EVALUATION OF BUS INFORMATION SYSTEMS

Jana Götze, Tatjana Scheffler, Roland Roller, and Norbert Reithinger

DFKI Projektbüro Berlin
Alt-Moabit 91c
10559 Berlin, Germany
firstname.lastname@dfki.de

ABSTRACT

In this paper, we describe our contribution to the Spoken Dialog Challenge. We set up a user simulation using the large Let's Go corpus as resource to build our models. Automatic calls were made to all four dialog systems in the SDC, bus information systems that cover the schedule of Pittsburgh, PA. We discuss in detail the architecture and required setup for our system-independent user simulation and report the results and challenges we faced. We also report initial evaluation results.

Index Terms— spoken dialog systems, user simulation

1. INTRODUCTION

User simulation is becoming an increasingly important part in the research on spoken dialog systems. The growing demand for systems that can be applied in various commercial or non-commercial domains calls for methods that facilitate their development and evaluation and user simulation can be a means for both. It helps in the training of models for the dialog manager and in evaluating the system as a whole.

As it is often difficult and time-consuming to test with human subjects, user simulation offers an alternative that comprises other advantages as well. A simulation's behavior is easier to control and can be modified at any given point. This allows for easy testing already during development. Another benefit for evaluation is the better comparability of results because the simulation changes its behavior only as specified by the developers and can thus be run many times. This is an important advantage over human testers who might behave in unforeseeable ways or get frustrated by the system limits. On the other hand, developers have to ensure that their simulation is able to cover as many features of human behavior as possible to obtain a natural simulation.

Currently, user simulation is mostly used for training a dialog system's dialog manager and building up resources for language and behavior models (see [1] for an overview of user simulation for SDS training) or for evaluating it at different stages of development. For both purposes a great variety of input is desirable to cover many different situations.

The SpeechEval user simulation [2] aims at exploiting corpora of human-machine dialogs to build a user simulation that is domain-independent and system-independent and communicates with a spoken dialog system for the purpose of evaluation. This framework allows for the automatic and semi-automatic creation of resources for the construction of a user simulation and provides utilities to use the simulated dialogs for the evaluation of a specified system.

The Spoken Dialog Challenge (SDC) organized by the Dialog Research Center of Carnegie Mellon University (DialRC) was set up to compare and evaluate different approaches in the development of SDS. This year's SDC was the first of its kind. The task was based on a deployed system, the Let's Go system, providing bus schedule information for the city of Pittsburgh, PA. Research groups were invited to adapt their system to the schedule task or to modify the Let's Go system with modules from their own research. The task specification thus left each group much freedom to decide to replace, change or extend the original system's architecture and capabilities. Besides the evaluation with human evaluators, the call was also open for systems in the area of user simulation and automatic evaluation. The SDC provides an ideal opportunity for us to test our simulation on realistic systems. The large Let's Go corpus of transcribed and annotated dialogs serves as an ideal resource to extract all the information that is necessary to set up our simulation. Since we developed our SpeechEval system based on a German dialog system corpus the SDC also was the trigger to create an English version of our simulation.

This paper is organized as follows. In section 2 we introduce the SpeechEval system and how it was used for the purpose of the SDC. Section 3 reports on our experiments and results, followed by a conclusion in section 4.

2. SPEECHEVAL USER SIMULATION

2.1. The General Approach

The SpeechEval user simulation was originally built for evaluating German spoken dialog systems. Its purpose is to serve as a general tool in the evaluation of SDS. Rather than special-

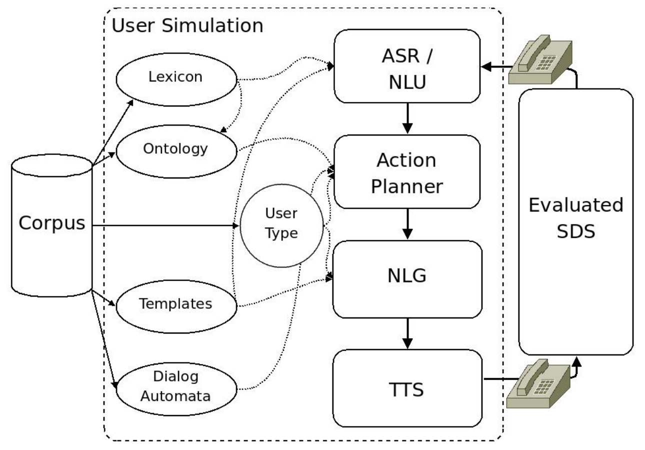

Fig. 1. Architecture of the SpeechEval system

izing on one particular system or domain the goal was to provide flexibility to cover a number of different domains. The main idea of SpeechEval is to learn as much as possible from existing corpora, in order to ease the evaluation designer from the tedious task of manually setting up all knowledge sources of the system.

SpeechEval was first developed and trained on a large corpus of human-computer dialogs from the Voice Awards[1] competition that annually evaluates commercially deployed SDS. The corpus contains about 2000 dialogs from 120 SDS [3]. This wide variety of different domains enables the simulation to use domain-independent as well as domain-dependent strategies in its dialogs. The collected dialogs were used to build a general recognition grammar, a model of user behavior and a collection of answer templates. Ontological information about the domain can partly be extracted from a corpus but has to be organized by hand. Depending on the task, it is also possible to specify extra information about the system that is to be tested, e.g. already known system prompts for the speech recognition.

The system's architecture (see fig. 1) maintains the modular organization of most SDS, consisting of modules for speech recognition (ASR), natural language understanding (NLU), action planning, answer generation (NLG) and a text-to-speech module. As our simulation uses speech as interface to the SDS a speech recognizer is the first part of the architecture and takes as input the SDS prompt received from the telephone line. Using speech instead of text or intentions as interface in the simulation has the advantages of being more realistic and more flexible. We also do not have to worry about introducing ASR errors in our output and our experiments show that the synthesized speech the simulation sends to the SDS is similar in recognition rate to human input (paper submitted).

[1]http://www.voiceaward.de

The NLU module is responsible for identifying the dialog act and additional keyword information in the incoming speech request. This module is closely connected to the ASR and mostly realized within the recognition grammar. The module's output are the recognized system prompt with its dialog act and keyword information. This information is passed on to the action planner.

The action planner constitutes the core module of every SDS and represents the user behavior of our simulation. It decides on what the system is going to do next. Depending on the analysis of the incoming prompt it chooses a user reply. Our answer planning is based on a statistical n-gram model that aims to match the probabilities of user behavior within the corpus. That means we model how people react to certain dialog acts and which kind of information they provide. We also consider communication errors like repetitions and misunderstandings by the SDS. This approach is close to the one used in [4] for an information state update system. Additionally we introduce different user groups (or stereotypes), based on various characteristics. This includes planning features such as cooperativity and average reaction time or generation features such as verbosity and fill word rate. Each stereotype has its own statistical model. The action planner's output is an answer dialog act and the information we want to reply.

The NLG module takes this information to generate a user reply. We have opted for a template-based approach, similar to [5]. Our templates are indexed for the dialog act they represent and the keyword information they contain. They also include a mark for length, ranging from templates containing only raw information to templates containing full sentences. The module selects an appropriate template based on the dialog act and keywords we want to provide and also takes into consideration its occurrence within the corpus and the current stereotype and situation. The template is then filled with information from the user goal and passed on to a speech synthesis which sends the signal back to the SDS via telephone.

All dialogs are logged with several features from the different modules and are later used for a usability evaluation.

2.2. Adaptions for the Spoken Dialog Challenge

The SDC was a good opportunity to test the user simulation for its usage in a real evaluation. The Let's Go corpus provided by the organizers was used to create the necessary resources for an English version of our user simulation that can communicate with the systems to be tested. The size of the corpus and the basic annotations are very well suited to set up a user simulation based on the core SpeechEval system.

To build a recognition grammar, it is possible to directly use all system utterances from the corpus. The corpus already contains annotation on the intention level in the form of dialog acts. For the NLU we could directly map dialog acts to system prompts and build up the recognition grammars.

system prompt	extracted slots
Going to <Children's Hospital> Did I get that right	ARRIVAL_PLACE
Leaving from <Forbes at Atwood> Is this correct	DEPARTURE_PLACE
There is a <56e> leaving <Murray Avenue at Hazelwood> at <8 42 p.m.>	ROUTE_NUMBER
	DEPARTURE_PLACE
	DEPARTURE_TIME
It will arrive at <Fifth Avenue at Grant> at <9 10 pm>	ARRIVAL_PLACE
	ARRIVAL_TIME

Table 1. example system prompts that were used to extract lists of named entities

When extracting the dialog act information for the SDS prompts, we decided to simplify the representation, mainly for reasons of consistency. The minor changes we made are renaming some of the acts (e.g. HOW_MAY_I_HELP_YOU into OPEN_QUESTION) and merging some acts where we judged the difference irrelevant to our simulation (e.g. LOOKING_UP_DATABASE and LOOKING_UP_DATABASE_SUBSEQUENT). To overcome the problems we had when extracting the information (see below) we automatically extended the annotation where possible. For example, any prompt of the form "[..] Did I get that right?" or "[..] Is this correct?" was annotated as EXPLICIT_CONFIRM.

The most frequently occurring system prompts are used to extract lists of named entities for the keywords that each system prompt contains, i.e. lists for the route numbers, the arrival and departure places and neighborhoods and certain system commands. See table 1 for some of the often occurring system prompts in the Let's Go corpus that we used for this extraction step. These lists are used to extract keyword information from a recognized prompt, such as determining that the system is trying to confirm the place of arrival.

At the core of our dialog management are the dialog act transition probabilities extracted from the corpus. Because of a tight time schedule we used a simplified user model that contained only bigrams extracted from the Let's Go corpus. Each unigram incorporates the dialog act and keyword information of an utterance. We also did not distinguish between different stereotypes.

Since the user prompts in the Let's Go corpus are not annotated with dialog act information, we use some of the parsing information the corpus contains for the user utterances. Parses are tagged as GENERIC_YES, GENERIC_NO or e.g. PLACE_INFORMATION and this information can be used to identify that the user is accepting or rejecting a system prompt or providing a certain piece of information. The named entities that have been extracted for the grammar are then used to identify the slots in these utterances, e.g. to determine that the user is conveying information about his departure time.

The NLG module takes the dialog act and keyword information to choose an appropriate template. For example, when given the information PROVIDE_INFO ARRIVAL_NEIGHBORHOOD the NLU module can choose one of the following templates:

- I'd like to go to <ARRIVAL_NEIGHBORHOOD>
- wanna go to <ARRIVAL_NEIGHBORHOOD>
- to <ARRIVAL_NEIGHBORHOOD>
- <ARRIVAL_NEIGHBORHOOD>

The template is then filled with the corresponding information from the user goal and sent to the text-to-speech module. The selection of templates is carried out randomly and not on the basis of their occurence within the corpus.

Initially, we used all user utterances that can be found in the Let's Go corpus for our templates, but first tests with the simulation revealed that the corpus is not as 'clean' as the VoiceAward corpus that was recorded under controlled conditions. The rather high number of strange utterances where callers obviously tried to trick the system required a manual clean-up before we could process the corpus.

The simulation also includes default actions. The dialog is automatically terminated when the system under test generates a prompt that it will hang up or that it is busy. We also hang up when the dialog exceeds a cut-off time, which is currently set to four minutes. Furthermore, we ignore input where our recognition confidence is too low but only if this doesn't happen more than three times in a row. If our simulation gets three SDS utterances in a row that are below our cut-off, we take the last one as valid input and ignore our cut-off until we get scores above it. For unknown input strings we look for keyword information and use these to generate a PROVIDE_INFO dialog act to tease out a follow-up clarification from the dialog system.

Overall, the data itself have proved an excellent resource, because they contained only real dialogs, i.e. all the callers called to obtain information for themselves and not because they were asked to do so. These dialogs were extremely useful when building a recognition grammar, user behavior models and templates for the generation of user responses. The data could also be used to get information on the domain, such as available bus routes and bus stops, although an exhaustive corpus would have been helpful in setting up user goals for the simulation.

departure place	forbes avenue at craig
departure neighborhood	oakland
arrival place	fifth avenue at sixth avenue
arrival neighborhood	downtown
route number	61 c
travel time	12 pm today

Table 2. An example goal for the user simulations

3. EXPERIMENTS AND RESULTS

In this section, we will report on our experiments (sections 3.1 and 3.2) and the results (section 3.3). Section 3.4 states the organizational and practical problems we encountered.

3.1. System Setup

Our experiments include calls to all of the four dialog systems, that are identified here as systems 1-4. We set up six different user goals with stops and neighborhoods extracted from our corpus. An example goal is shown in table 2. Each goal included the stop and neighborhood of both departure and arrival, the time of departure and a bus line. Where we could not match a neighborhood or route number from the corpus we used the online travel information system provided by Pittburgh's Port Authority[2].

Our simulation was set up to call each system five times for each of the six goals. After making some adjustments to the ASR we carried out another run. This should have yielded a total of 30 dialogs per system, 120 dialogs altogether. However, for various reasons (see subsection 3.4) we obtained a total of 111 dialogs. An extract from a dialog can be found in table 4.

For the experiments, Nuance version 9.0 is used for our speech recognition and MARY TTS as speech synthesis (voice cmu-slt-hsmm). We report our average confidence scores for the ASR because the simulation results may be influenced by them. The differences in confidence scores between the systems may be due to different telephone connections and different synthesis methods (see also section 3.4).

3.2. Method

Of the 111 dialogs that we carried out with our user simulation, only 48 are suitable for comparison. There are a number of dialogs where the user hangs up. Our model of user actions is directly derived from the corpus and contains all user actions. Since users may have chosen to hang up at any point during the dialog for reasons that we do not know, the model also contains these actions and our simulation can make use of them. Note that the overall task success rate in the corpus is also rather low (54.95% disregarding dialogs where the user didn't say anything) for this reason. For the purpose of evaluation these dialogs are not suitable because we cannot tell why a dialog was terminated. Our results thus only report on dialogs where the SDS could find a result or where the user simulation hangs up after four minutes.

[2]http://www.portauthority.org

For a first evaluation of the dialogs that the systems produced with our user simulation, we choose to report some simple measures that can be extracted from our dialog logs.

- average number of turns per dialog (*tpd*) to capture the time it takes to complete the dialog. This is measured as the number of turns of both the simulation and the system divided by the number of completed dialogs.
- average turn length (*atl*) in seconds to put the number of turns per dialog into perspective.
- task completion rate (*tcr*) to convey how often a query was successful. This value is currently counted as the number of times the system outputs a result. The simulation hangs up on the system per default after obtaining a result, so a dialog contains at most one query.
- the rate of explicit confirms over all confirms (*ex-conf*) to capture how well the system identifies the user's information. Using more implicit confirms than explicit confirms can be an indication for high confidence in understanding the user.
- understanding rate (*ur*) to measure how many turns the system needs to collect a piece of information. This is measured for the arrival and departure places and the travel time. We leave out the route number because a system does not need it to find a result.

3.3. Results

We report the measures for all tested systems in table 3.3 but leave out one of them (system 1) when comparing them, because unfortunately we could only obtain four dialogs and none of them was successful. Our confidence for the other three systems are nearly equal (.55 for both systems 2 and 4, .57 for system 3). For our lack of sufficient data we will not give a full evaluation here but only remark on some issues that we found and that are reflected in the measured values.

System 2 has the highest task completion rate, it could return a result bus route in 17 out of 18 dialogs (94%). It needed on average 26.59 turns to complete a dialog. On the other hand, system 4 has the lowest task completion rate (only 8 out of 13 dialogs (62%) yielded a bus route) but also needs the fewest number of turns to complete its task (13.25). The completion rate of system 3 is 71% but the average turn number per dialog is even a little higher than that of system 2 (28.17).

The high number of turns per successful dialog in system 3 is also reflected in our understanding rate. The system takes many turns to understand and verify the different pieces of information, especially the departure and arrival places. This can be due to different reasons. The system's ASR might have had problems understanding our TTS, e.g. because of a bad telephone connection. The system might also have had trouble with only one certain piece of information and our simulation wasn't able to rephrase it in a way for the SDS to understand. The user goals were the same for all four systems, but the dialogs should also be further evaluated for the impact of the different goals to find out whether some place names are harder to understand than others.

	sys1	**sys2**	**sys3**	**sys4**
# of dialogs	4	18	17	13
acs	0.79	0.55	0.57	0.55
tpd		26.59	28.17	13.25
atl	4.22	6.31	6.03	5.31
tcr	0.0	0.94	0.71	0.62
ex-conf	1.0	0.60	1.0	1.0
ur				
departure		2.08	2.79	1.80
arrival		1.95	3.36	2.13
travel time		2.10	1.56	1.00

Table 3. The experiment results: the number of recorded dialogs, the average confidence score (acs), the turns per dialog (tpd), the average turn length in seconds (atl), the task completion rate (tcr), the rate of explicit confirms (ex-conf) over all confirms and the understanding rate for different items (ur)

All systems make different use of confirmations. The only system that uses implicit confirms is system 2. About 40% of all confirmations are implicit and this has proved difficult for our simulation (see section 3.4). The impact of confirms on usability and quality is not fully understood. They are a means of clarifying input where the system is not sure enough to proceed but can be a source of frustration for the user if used too often. Implicit confirms are usually a means of conveying what was understood but still giving the user the possibility to intervene. A reasonable alternation between explicit and implicit confirms, taking into account recognition confidence scores may prove a good strategy, but this an open research question.

The understanding rate, defined as the number of turns a system needs to collect a certain piece of information, can give some indication on the system's ability to understand what the user is trying to say. There is no clear indication of one piece of information being harder to understand than another. There should be no difference between departure and arrival place because both are taken from the same set of places. The results also don't show an overall difference between the places and the time information. System 4 was able to collect the travel time fastest with only asking for it once, for system 2 it seemed to take longest to understand the time information. When looking more closely at the dialogs, the reason for this is that the system splits up the information by asking separately for the day and time of travel. The results for system 3 suggest that it had some problems recognizing place information as it took about three turns to understand both arrival and departure place.

Based on these few measures it is not reasonable to draw a valid conclusion on the differences in quality between the systems. It remains to be seen how the human evaluations relate to our simulation measures.

3.4. Experiences during the Tests

The SDC was organized according to a fixed timeline for the development of the systems, their deployment and the evaluation procedure. Nevertheless, the schedule left room for the participants to state and discuss their own opinions on issues like the logging of information. As this year's SDC was the first for both organizers and participants and since there was no best practice yet, the participating research groups collaborated very closely during the preparation and evaluation phase. The SDC included user simulations for evaluation from the beginning. However, since this type of evaluating evaluation systems was rather new, the criteria for evaluations had to be defined. Continuous communication and discussion with the groups developing the systems helped us decide on solutions that we judged reasonable for everyone, but a more fixed framework could help overcome such formal problems.

While we carried out the adaption and test of the simulation, we encountered a number of problems. Most of them are related to the speech recognition, but there were also problems concerning the connection to the systems. This was either due to database problems or miscommunication about the times when the systems would be running.

For the first runs of the simulated dialogs, our recognition confidence was very low. This was due to various factors. First of all, it was related to the structure of our grammar. The recognition was carried out dialog act by dialog act. As it turned out, in some systems the different dialog acts followed in quite short intervals (e.g. system 2 uses implicit confirms, followed by instructions and requests). During the calculation and output of a user reply, the recognition is switched off for a short time. If during that time we receive the next speech input, we miss its beginning. We had to restructure our grammar to solve this issue. A modification of the recognizer settings and an optimization of the streaming from telephone to ASR have also improved the recognition.

Second, we extracted the place names from the corpus and thus do not have a complete list. This results in a lower confidence whenever the SDS uses one of the out-of-grammar place names. For the other sentences that are recognized with a low confidence we probably have to optimize the pronunci-

speaker	utterance
sds	Where would you like to leave from?
user	From Oakland.
sds	What is your destination?
user	Downtown.
sds	When do you want to travel?
user	A bus at 12 54.
sds	Travelling at 12 54. Is this correct?
user	Alright yes.
sds	Travelling at 12 54 p.m. Is this correct?
user	Correct.
sds	Leaving from the Oakland neighborhood. Is this correct?
user	That's right.

Table 4. An extract from a simulated dialog (system 4)

ation dictionaries of our ASR.

Another problem seemed to be the precise domain of the SDS we tested. We encountered the situation that system 3 was trying to suggest to our simulation a place to eat. We also had the case where instead of a place name there was only noise in the SDS output. This might also have been due to a technical problem at the system's site, e.g. a database connection problem.

Since the SDC was open for systems around the world, we also encountered a very human problem: different time zones for the systems to be tested. The American systems were up and running at night in Germany, so we had to set up our simulations partially unsupervised. In case of technical problems, like system crashes or unavailable systems, we could only intervene the next morning. So there are still some technical issues open we could not identify.

4. CONCLUSION

This paper has demonstrated how a new corpus can serve as a resource for the setup of our SpeechEval user simulation. The Let's Go corpus was ideal for extracting all the necessary information to set up models for the different modules. This has proved easy even for a new language.

The user simulation tests have revealed a number of technical issues that need to be overcome to get meaningful results. However, the generated dialogs can serve as a basis for further improvement in our simulation strategy and be a resource for next year's SDC. Although this corpus of simulated dialogs is small, it can also be used to get a first insight into other interesting questions, such as the differences in the English versus German behavior models (in-domain and across-domain).

In this year's SDC, we only participated with SpeechEval's user simulation. The second part of SpeechEval, the usability evaluation of the automatically tested dialog systems, was not used to analyze the test runs due to the tight schedule. As a follow-up acitivity we will also test the automatic usability evaluation on the data gained from the SDC. It will also be interesting to see how the tests with human users relate to our findings and how other user simulations perform in future Spoken Dialog Challenges.

Overall, the Spoken Dialog Challenge was a very inspring and, yes, challenging contest. At least for our simulation we gained a tremendous amount of experience.

5. ACKNOWLEDGEMENTS

We would like to thank the organizers and the other participants in the SDC for their very cooperative collaboration.

This work is part of the project SpeechEval: Automatic Evaluation of Interactive Speech-Based Services on the Basis of Learned User Models, which is being carried out in cooperation with the Quality and Usability Lab at Technical University Berlin. SpeechEval is funded by the Investitionsbank Berlin through the ProFIT framework, grant #10140648. This project is being co-financed by the European Union (European Regional Development Fund).

6. 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," *The Knowledge Engineering Review*, 2006.

[2] Sebastian Möller, Robert Schleicher, Dmitry Butenkov, Klaus-Peter Engelbrecht, Florian Gödde, Tatjana Scheffler, Roland Roller, and Norbert Reithinger, "Usability engineering for spoken dialogue systems via statistical user models," in *First International Workshop on Spoken Dialogue Systems Technology (IWSDS 2009)*, Kloster Irsee, Germany, December 2009.

[3] Tatjana Scheffler, Roland Roller, and Norbert Reithinger, "Semi-automatic creation of resources for spoken dialog systems," in *KI 2009: Advances in Artificial Intelligence*, B. Mertsching, M. Hund, and Z. Aziz, Eds. 2009, vol. 5803 of *LNAI*, pp. 209–216, Springer.

[4] K. Georgila, J. Henderson, and O. Lemon, "Learning user simulations for information state update dialogue systems," in *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech)*, Lisbon, Portugal, 2005.

[5] R. López-Cózar, A. de la Torre, J. Segura, and A. Rubio, "Assessment of dialog systems by means of a new simulation technique," *Speech Communication*, vol. 40, pp. 387–407, 2003.