# Doing Research on a Deployed Spoken Dialogue System: One Year of Let's Go! Experience

*Antoine Raux, Dan Bohus, Brian Langner, Alan W Black, Maxine Eskenazi*

Language Technologies Institute
Carnegie Mellon University, Pittsburgh, USA
{antoine, dbohus, blangner, awb, max}@cs.cmu.edu

## ABSTRACT

This paper describes our work with Let's Go, a telephone-based bus schedule information system that has been in use by the Pittsburgh population since March 2005. Results from several studies show that while task success correlates strongly with speech recognition accuracy, other aspects of dialogue such as turn-taking, the set of error recovery strategies, and the initiative style also significantly impact system performance and user behavior.
**Index Terms**: spoken dialogue systems, real-world applications, speech recognition

## 1 INTRODUCTION

Much has been achieved by the spoken dialogue systems research community in the past decade. Systems handle more and more advanced tasks and accept increasingly complex natural language input. Unfortunately, such research is usually conducted on "toy" systems where users are usually students given scenarios to follow. On the other hand, little research is published on real world systems catering wide user populations, such as commercial IVR systems.

Fundamental research needs to be carried out on a system that has real users in large numbers to be validated. The Let's Go Bus Information System bridges this gap [1]. It was created at Carnegie Mellon, using the RavenClaw dialogue manager [2], the Sphinx 2 speech recognition engine and a domain-specific voice built with the Festival/Festvox toolkit and deployed on the Cepstral Swift engine [3].

Let's Go gives bus schedule information to the Pittsburgh population at hours when the Port Authority phones are not manned by operators (7pm to 6am on weekdays and 6pm to 8am on weekends). Having collected some 20,000 calls in the year that it has been up and running (since early March 2005), it furnishes a great platform for spoken dialogue research.

## 2 CALL TRAFFIC AND OVERALL PERFORMANCE

### 2.1 Call traffic

The average number of calls reaching Let's Go! is about 40 on weeknights and 60 on weekend nights. Average daily call traffic for the past year has oscillated between 40 and 60 (see Figure 1), depending on seasonal variations and special events such as schedule changes (including a major reorganization in September which boosted call traffic for the next two months)[1]. We also found that the average daily traffic for March 2006 was about 10% higher than for March 2005, which might indicate a certain proportion of return users[2].

The average length of a dialogue is 14 turns. However the distribution of dialogue turn lengths, shown in Figure 2, is bi-modal, with a first peak at 0 turns (10% of the dialogues) and a second one around 10 turns. This reflects two types of user behavior, with part of the population hardly trying to use the system at all and the others spending more time to attempt to get their information. The minimum number of turns to successfully get schedule information, given all the necessary confirmations is 6. These observations lead us to exclude short dialogues (less than 6 turns) from statistics in this paper, since they might not be genuine attempts at using the system.

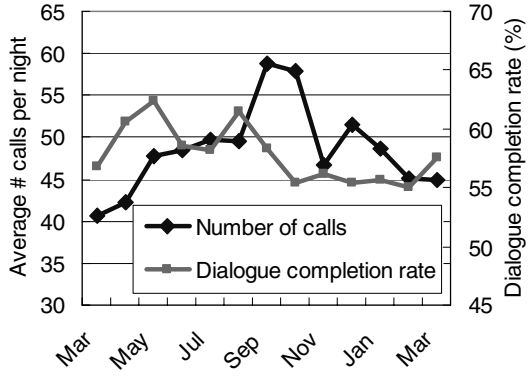


**Figure 1 Average daily call traffic and dialogue completion rate (for dialogues with 6 turns or more) from March 2005 to March 2006.**

[1] In November, technical issues caused the system to drop some calls, resulting in the observed lower number.
[2] Prior to the deployment of Let's Go, callers would simply get a recorded message telling them to call back during regular business hours.

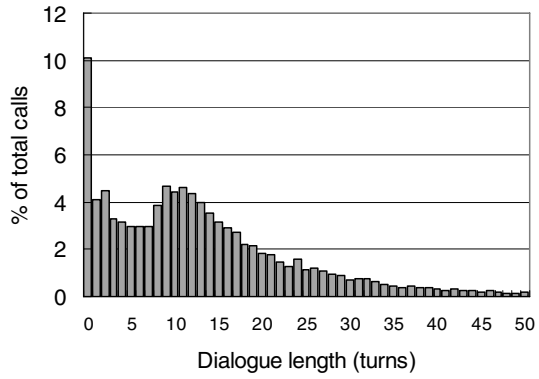


**Figure 2 Distribution of dialogue length**

### 2.2 Dialogue completion rate

In order to have a running estimate of success rate without having to manually label each call, we used the following heuristic: a dialogue is marked as complete if the system obtained enough information from the user to either provide a result (by performing a database look-up) or notify the user that their request is out of coverage (e.g. it is for a bus route that we do not deal with yet). We call the ratio of complete calls to the total number of calls "dialogue completion rate" (DCR).

By manually labeling task success for a subset of the calls (based on recordings), we found that complete dialogues have a 79% success rate. By construct, incomplete dialogues are necessarily failures (since no information was given to the user). Therefore an estimate of task success rate could be derived from DCR. In this paper, we only report task success rate when dialogues were manually labeled. In all other cases, we report dialogue completion rate. The evolution of DCR over the past year is shown in Figure 1. Overall, DCR has remained stable, variations being due to chance and to the various experiments we conducted during the year.

## 3 FACTORS AFFECTING PERFORMANCE

### 3.1 Speech Recognition Accuracy

The accuracy of the speech recognizer is often considered the single most important factor in making a successful speech application. To analyze this aspect of Let's Go, we transcribed the calls from March 2005, which represent 586 dialogues and 9104 utterances. We report the session-average word error rate (SA-WER), which is computed by averaging the WER of all the turns in a session. The average SA-WER overall is 64.3%. Figure 3 shows hand labeled task success as a function of SA-WER. Except for the left-most point where too little data was available, it can be noted that there is a strong linear correlation between the two ($R^2$=0.97 for WER over 10%).

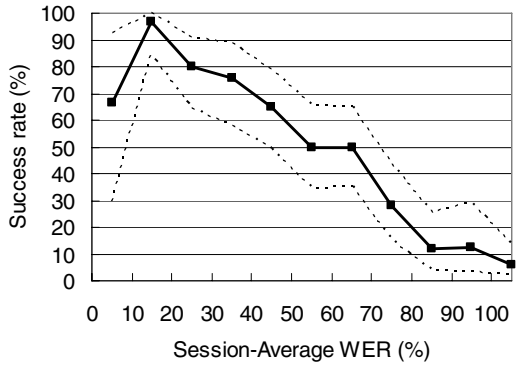


**Figure 3 Success rate as a function of WER.** The dotted lines represent 95% confidence intervals.

In an attempt to improve the system's performance, we retrained our acoustic models by performing Baum-Welch optimization on the transcribed data (starting from our original models). Unfortunately, this only brought marginal improvement to the WER (computed on a left-out test set). We suspect that this lack of improvement is due to the fact that the models (semi-continuous HMMs) and algorithms we are using are too simplistic for this task. In the near future, we will investigate other recognition engines and a fully train model on collected Let's Go data

### 3.2 Turn-Taking

Let's Go, as most current dialogue systems, relies on an energy-based endpointer to identify user turn boundaries. Although barge-in from the user is allowed at certain points in the dialogue, the system does not have rich turn-taking management abilities and therefore imposes a fairly rigid model of turn-taking to the dialogue. For example, the user backchanneling to the system is often misinterpreted as a barge-in and leads to the system interrupting itself when it should not. Also, if the user introduces long pauses within her utterances, they are likely to be misinterpreted as turn boundaries, leading to confusion and misrecognitions. In order to quantify the amount of turn-taking issues and their impact on the dialogue, we inspected a subset of our calls and manually labeled five types of turn-taking failures (see Table 1). Unfortunately, because it is necessary to listen to the whole dialogue, this labeling is a fairly expensive task and we could only label 102 dialogues.

For each failure, we counted the proportion of dialogues in which it occurred. The most frequent failure is when the system misinterprets a noise or a backchannel from the user as a barge-in and wrongly interrupts its current turn. While we disabled barge-in on crucial turns (e.g. when giving the results of the query), we still allow the user to barge in at many points in the dialogue. While this allows a swifter interaction for expert users, it has a significant cost as this failure appeared in more than half of the dialogues (52%). Next in frequency (47%) is the system failing to take a turn, usually due to inaccurate endpointing (e.g. the system does not endpoint because of background noise). Third is the converse of the first one, namely the

system failing to interrupt itself when the user actually attempts to barge in. This generally happens on prompts where barge-in is intentionally turned off, and shows that this option is not optimal either since user can get frustrated if the system does not respond in a timely fashion. Finally the last two failures occur when the system starts speaking when it should not, right after its own turn ("System takes extra turn"), usually due to a noise misinterpreted as a user turn, or in the middle of a user turn, usually by misinterpreting a hesitation pause with a turn boundary.

**Table 1 Frequency of occurrence of five turn-taking failures**

| failure type | frequency (% calls) |
|---|---|
| System wrongly interrupts its turn | 52.0% |
| System fails to take a turn | 47.1% |
| System fails to yield a turn on user barge-in | 43.1% |
| System takes extra turn | 39.2% |
| System wrongly barges in on user | 15.7% |

Overall, turn-taking failures occurred more frequently than we had anticipated, with 85% of the calls containing at least one such failure and, on average 3.8 failures per call. In addition, inspection of the dialogues showed that 10% of them broke down mainly for turn-taking reasons, which represents about 20% of the failed dialogues. While we need more data annotation and analysis to draw definitive conclusions, this study shows that there is much room for improvement at the turn-taking level. In the short term, we will focus on making the endpointer more robust by relying on more features than just energy to detect turn boundaries. In the longer term, our research endeavor will aim at building a flexible turn-taking model for dialogue systems; a feature which we feel is badly needed not only in Let's Go but in other systems as well.

### 3.3 Non-understanding recovery strategies

Another way to improve the performance of a dialogue system is by improving its error handling capabilities.. Let's Go is based on the RavenClaw dialogue manager [2], which, among many other features, provides a set of domain-independent dialogue strategies for handling non-understandings. The initial set of strategies was designed based on our intuition and our experience with research spoken dialogue systems. This original set is described in Table 2. In early 2006, having learned a lot from almost a year of experience with a real-world system, we modified the set of non-understanding recovery strategies (see Table 2). The modifications were of three types: rewording of system prompts, removal of ineffective strategies, and addition of new strategies.

Our experience with Let's Go suggested that long prompts were not well received by the users and were mostly ineffective. Consequently, we removed non-critical informational content from prompts, shortened them, and made them as specific as possible. For example, many prompts started with a notification that a non-understanding had occurred ("Sorry, I didn't catch that.").

**Table 2 Non-understanding recovery strategies in the old and new version of Let's Go!** (*: the prompt for this strategy was preceded by a notification prompt)

| strategy | old | new |
|---|---|---|
| General help on how to use the system | X* | X |
| Local help + context-dependent examples | X* | |
| Context-dependent examples | X* | X |
| General help + local help + c.-d. examples | X* | |
| Give up question and go on with dialogue | X | X |
| Repeat question | X* | X |
| Ask user to repeat what they said | X | X |
| Ask user to rephrase what they said | X | X |
| Ask user to go to a quiet place | | X |
| Ask user to speak louder | | X |
| Ask user to use short utterances | | X |
| Offer to start over | | X |
| Give up dialogue and hang up | | X |

During such prompts, users would frequently barge in on the system right after the notification and thus not hear the following utterance, which contained help or examples of expected user utterances. We therefore decided to eliminate the notification prompt so that the user could hear the more informative specific content of each prompt.

We also removed generic help prompts, which explain what the system is trying to do at a given point, since they didn't appear to help much. However, we kept the help prompts giving example utterances, which were more often picked up by users and were thus more effective.

Finally, we added more specific strategies, aiming at dealing with problems like noisy environments, too loud or too long utterances, etc. The idea was that such pinpointed strategies, if used at the right time, would be more effective in addressing the issues hurting communication between the user and the system. Currently we use simple heuristics to trigger each of these strategies, based for example on the length of the last user utterance or a simple audio clipping detection algorithm.

To evaluate the impact of these changes, we manually labeled the action following each non-understanding as successful when the next user utterance was correctly understood by the system or failed when the next user utterance led to another non-understanding or to a misunderstanding. We labeled three days of data before the modifications took place and three days after. We used the same days of the week (three weeks apart) to mitigate the effect of daily variations in performance. The results indicate that the modifications significantly improved the success rate of non-understanding recovery strategies, from 19.8% to 24.1% ($p<0.01$). The overall dialogue completion rate also went up from 49.7% to 55.2% although this result is only a trend ($p<0.1$).

We are currently performing more analyses to better understand each strategy's performance and to design a mechanism to learn from data when to trigger each strategy so as to optimize its performance.

# 4 TOWARDS MIXED INITIATIVE

## 4.1 Closed vs open prompt

While Let's Go! was originally designed to be strongly system-directed, we wanted to investigate the impact of varying both prompt wording and initiative style. To limit the variability introduced in the system and avoid hurting performance too much, we only modified the initial prompt of the system. We created three versions of the system, with the following initial prompts:

1) "Which bus number or departure place do you want information for?"
2) "What bus information are you looking for?"
3) "What can I do for you?"

In version 1, the system only recognized bus numbers and places at this point in the dialogue, whereas in version 2 and 3, the system could understand more general utterances such as "When is the next bus from CMU to downtown?". If the system failed to understand anything on the user's first utterance, it gave a help message with examples of appropriate utterances (the examples were different for version 1 vs 2 an 3). After a second non-understanding, the system would always fall back to version 1. Thus the only differences between the versions are in the first turns of the dialogue.

We let the system run with these three versions in August and September 2005, collecting around 1000 dialogues for each condition (see Table 3). No manual transcription or labeling of the data was performed, thus all reported results are based on automatically extracted measures.

## 4.2 Difference in user behavior

Our hypothesis in terms of user behavior was that version 1 would yield short, specific answers, whereas version 3 would yield the longest and most diverse responses, with version 2 lying in the middle. Table 3 shows the average duration of the first utterance of the dialogue.

As expected, the most "open" prompt (version 3) leads to the longest answers (2.83s on average), whereas the most "close" leads to the shortest (1.75s). Interestingly, the intermediate version yields a behavior very close to version 1 (the difference is not statistically significant). As a possible explanation for this last similarity, we found that many users replied to this question by a single bus route number, indicating that they understood the question not as an open prompts but rather as meaning "which bus route's information are you looking for?"

**Table 3 Impact of initial prompt's initiative style on user behavior and system performance**

(DCR=Dialogue Completion Rate, NUR=Non-Underst. Rate)

| system version | # calls | duration (ms) | NUR (1st utt.) | DCR | avg # turns |
|---|---|---|---|---|---|
| 1 | 1063 | 1678 | 32.2% | 54.9% | 17.9 |
| 2 | 1006 | 1750 | 28.9% | 52.4% | 17.5 |
| 3 | 999 | 2828 | 52.9% | 51.5% | 18.2 |

## 4.3 Difference in system performance

We also analyzed the impact of initiative style on system performance, expecting that more open user utterances would be harder to recognize and understand and therefore lead to lower system performance. First we computed non-understanding rate for the very first user utterance of the dialogue, i.e. the proportion of dialogues where no information could be extracted by the system from the first user utterance. These results, shown in Table 3, conform to our expectations, in that the non-understanding rate is significantly higher (52.9%) for version 3 than it is for either version 1 or version 2 (resp. 32.2% and 28.9%). On the other hand, no statistically significant difference was measured for task success nor for the average length of successful dialogues.

# 5 CONCLUSION

We intend to continue to run the system for the public at large and have started investigating how to use the live system for wide variety of experiments on dialogue. Due to the large number of real calls compared to most research systems, we feel Let's Go! is an excellent platform for experimentation and evaluation. We are devising a Let's Go Lab that will provide a well-defined mechanism to the community for running selected experiments on this system.

# 6 ACKNOWLEDGEMENTS

This work is supported by the US National Science Foundation under grant number 0208835, "LET'S GO: improved speech interfaces for the general public". Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The authors would like to thank the Port Authority of Allegheny County for providing access to their database and for their help in making the Let's Go system accessible to Pittsburghers.

# 7 REFERENCES

1 Raux, A., Langner, B., Bohus, D., Black, A., Eskenazi, M. *Let's Go Public! Taking a Spoken Dialog System to the Real World*, Interspeech 2005 (Eurospeech), Lisbon, Portugal, 2005.

2 Bohus, D., and Rudnicky, A., *RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda*, Eurospeech 2003, Geneva, Switzerland, 2003.

3 Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y., Lee, K.-F. and Rosenfeld, R., *The SPHINX-II Speech Recognition System: an overview*, Computer Speech and Language, 7(2), pp 137-148, 1992.

4 Black, A. and Lenzo, K., *Building Voices in the Festival Speech System*, http://festvox.org/bsv/, 2000.

5 Cepstrall, LLC, *SwiftTM: Small Footprint Text-to-Speech Synthesizer*, http://www.cepstral.com, 2005.