What You Did and Didn't Mean: Noise, Context, and Human Skill

Tiziana Ligorio (tligorio@gc.cuny.edu)
Susan L. Epstein (susan.epstein@hunter.cuny.edu)
Department of Computer Science, Hunter College and The Graduate Center of The City University of New York
New York, NY 10016 USA

Rebecca J. Passonneau (becky@cs.columbia.edu)
Joshua B. Gordon (gordon@cs.columbia.edu)
Center for Computational Learning Systems, Columbia University
New York, NY 10037 USA

Abstract

In spoken dialogue between people and machines, the computer must understand not only what the speaker means but also what she does not. The computer begins with a considerable disadvantage: even the best speech recognition technology can provide error-ridden transcriptions of human speech under real-world telephone conditions. The work recounted here examines how, and how well, people use context to interpret noisy transcribed utterances in a challenging domain. Models learned from this experiment highlight two aspects of this human skill: the ability to detect a context-supported match, and the ability to know when the quality of attempted matches is so poor that it should be questioned. These models can then be applied by a spoken dialogue system to find the correct interpretation of users' spoken requests, despite incorrect speech recognition.

Keywords: spoken dialogue systems; natural language processing; machine learning; Wizard of Oz studies, learning.

Introduction

A computer system intended to replicate a human skill faces two considerable limitations: it works from a different input modality and it is restricted to a preprogrammed set of alternative actions. The thesis of our work is that people should be studied for their skill at the target task as if they were similarly restricted, that is, as if they had only the system's input and alternatives. The domain of investigation here is a spoken dialogue system (*SDS*). Subjects were given the same data that would be available to the SDS: error-ridden strings representing transcribed speech plus a large database of possible matches (Passonneau et al., In Press). The resultant data was then used to identify the best performers, and to learn models of them destined for the system. There were two principal results. First, subjects ably guessed what the speaker meant, that is, they could often identify the correct item from the context provided by a database query on the error-ridden string. Second, the people most skilled at this task excelled because they could also identify what the speaker had *not* meant, that is, they knew when no item returned from the database was a correct match. Such recognition is essential to move dialogue forward constructively

Ideally, an SDS offers people a natural way to communicate with a computer and benefit from its expertise. In an SDS, automated speech recognition (*ASR*) transcribes human spoken input into a string of words, which is then assigned an interpretation. Under real-world telephone conditions, however, even state-of-the-art ASR can exhibit a word error rate (*WER*) as high as 68% (Raux et al., 2005). High WER is common when the environment is noisy, the language the system is expected to understand is flexible and based on a large vocabulary, or the user population is diverse in gender, age, and native language. These are all characteristic of our target domain: telephoned book requests from patrons of the Andrew Heiskell Braille and Talking Books Library.

Although people manage dialogue well in the presence of noise, computers do not. Our subjects used ASR output from a spoken title to query our copy of Heiskell's book database. For example, for one book title, the ASR output string was "ROLL DWELL." A database query on this string returned three likely matches (in real time): "CROMWELL," "ROBERT LOWELL," and "ROAD TO WEALTH."

This is a difficult task. (The reader is invited to guess whether any of these actually matches the spoken title.) Our experiment studies how people manage this task. The resultant data is then used to train accurate models of human behavior, and to identify the features that make people proficient. Such models are ultimately intended as an integral part of an SDS, to make it more robust to noisy ASR in the context of database queries. The next sections of this paper describe related work, our target system, and the experimental design and results. These are followed by a description of the models learned from the data collected during the experiment, and a discussion of their import and application.

Related Work

The Wizard of Oz (*WOz*) paradigm is a well-known approach to iterative prototype design. It gathers information about the characteristics of a successful system before the system's development (Dix, et al. 2003). In a WOz experiment, only a user-system interface is provided. Users believe they are interacting with a computer system through this interface, but instead a person (the *wizard*) is "behind the curtain." This permits the system designer to observe human responses to certain system functionalities, to study user behavior and expectation, and to assess interface design features before the construction of an initial prototype.

WOz can also be used to study the wizard, to provide data on how the system should behave. In particular, *wizard ab-*

lation is a WOz study in which a wizard relies on system input and output rather than her own communication resources (Levin and Passonneau 2006). Wizard ablation supports the collection of dialogues that illustrate the decisions people make when confronted by the same input/output data and choices as an SDS. Data collected under wizard ablation supports supervised learning to predict wizard actions. The resultant model can then be incorporated into a system to improve its behavior. WOz studies that directed their attention to the wizard during full spoken dialogues include efforts to predict: the wizard's response when the user is not understood (Bohus 2004), the wizard's use of multimodal clarification strategies (Rieser and Lemon 2006), and the wizard's use of application-specific clarification strategies (Skantze 2005). The experiment presented here is restricted to single utterances, rather than full dialogues. It also differs in that it analyzes several wizards' behavior. It recognizes differences among wizards and identifies distinctive and successful behavior, so that the system will ultimately benefit only from models of the most skilled wizards.

To limit communication errors incurred by faulty ASR, an SDS may use enriched strategies to detect and respond to incorrect recognition output (Bohus 2004). It may repeatedly request user confirmation to avoid misunderstanding, or ask for confirmation using language that elicits responses from the user that the system can handle (Raux and Eskenazi 2004). When the user adds new information in response to a system prompt, two-pass recognition can consider the extra information contained in such user responses to restrict the language expected in the second pass and thereby achieve better recognition (Stoyanchev and Stent 2009). In a highly interactive setting, an SDS might benefit when it takes this approach one step further and uses context-specific language for incremental understanding of the noisy input throughout the dialog (Aist, et al. 2007). This paper explores the use of system-internal resources, such as a database search, to respond to faulty ASR. It embeds a wizard into a system, and then observes and models her ability to use such context to respond appropriately.

Peripherally related are other approaches that increase understanding between an SDS and the user through the adaptation of an SDS's response based on a user model. In automated tutoring, for example, it is essential to validate the user when she is correct and to elicit more reasoning when she is not (Franceschetti, et al. 2003, Ohlsson, et al. 2003). In particular, affect-adaptive systems can improve learning efficiency by responding to uncertainty in the transcribed speech (Forbes-Riley and Litman 2009).

Ordering Books with CheckItOut

CheckItOut is a research SDS for book requests from patrons of the Andrew Heiskell Braille and Talking Books Library, a branch of the New York Public Library and part of the National Library System. Patrons of the library request books by telephone and receive them by mail. Regular newsletters provide patrons with the titles and catalogue numbers of new books. To gauge the kinds of interactions patrons have with Heiskell's librarians, we transcribed 82 telephone calls from a larger set we had recorded. Forty four percent of the book requests were by catalogue number, 28% by title or a combination of title and author, and 28% were more general. CheckItOut is therefore designed to accept book requests by catalogue number, author, or title.

CheckItOut builds upon the Olympus/Ravenclaw architecture and dialogue management framework (Bohus, et al. 2007, Bohus and Rudniky 2003). Olympus/Ravenclaw has been the basis for approximately a dozen research SDSs in different domains. During a dialogue with CheckItOut, the user first identifies herself as a patron of the library, and then requests at most four books. CheckItOut references two databases: a sanitized version of Heiskell's database of 5,028 patrons, and its entire book database with 71,166 titles and 28,031 authors. These force CheckItOut to manage a large vocabulary; titles and author names alone contribute 54,448 distinct words. Moreover, Heiskell's patrons include many elderly and non-native speakers. The experiment described next observes how human wizards respond to the same challenges that CheckItOut confronts.

Experimental Design

The experiment described here seeks to uncover how people marshal system resources (e.g., the ASR string and database results), and which strategies achieve the best performance. Here the focus is on single turn interactions that request books by title, the CheckItOut request type most likely to elicit problematic ASR output.

In an offline pilot study, 3 native speakers of English read 50 titles to generate 3 sets of ASR output strings (Passonneau, et al. 2009). Each subject received a different ASR set and was asked to find the corresponding title from a text file that listed all 71,166 titles. WER was 69% – 83%, depending on the speaker. Despite the high WER, these subjects identified the correct title 74% of the time.

Given this demonstration of human skill, we designed a WOz study to identify which aspects of human performance come into play when a wizard seeks to match noisy ASR against a list of *candidates* (possible title matches) (Passonneau et al. In Press). The experiment was designed to identify what makes a good wizard, and to extract any additional insights a wizard may offer when supported by database search with the quality common in modern systems.

During the experiment, users and wizards were isolated from one another in separate rooms. Each had her own graphical user interface (*GUI*) and microphone. In a *title cycle*, the user read a book title into a speech recognizer through the microphone, and the corresponding ASR was displayed on the wizard's GUI. The wizard then formulated a query for the database. Once the search returned a list of candidates, the wizard had four options: make a *confident choice* among the candidates, make a *tentative choice* among the candidates, ask a *question* through her microphone, or *give up*. (Wizards were also permitted to ask the user to repeat the title, but were discouraged from doing so.) If the wizard chose a candidate, it then appeared on the us-

user's GUI, and the user scored it as correct or incorrect. That score was also displayed on the wizard's GUI, so that the wizard knew if her most recent title choice was correct or incorrect. If the wizard asked a question instead, the user heard it through her headset and rated it on her GUI. The possible ratings with respect to the current book request were "relevant and I can answer it," "relevant but I cannot answer it," irrelevant," and "uncertain." Question ratings were not shared with the wizard. After the wizard saw the user's score or was notified that the user had judged the question, the wizard signaled the beginning of a new cycle.

The speech recognizer continually transcribed the speech signal from the user's microphone, and the wizard's GUI provided a live feed of the resultant ASR strings. For each request, the wizard submitted a database query after very limited editing of those strings (e.g., removing "um"). The return from the database was displayed on the wizard's GUI as a list of candidates in descending order of search confidence. This confidence was measured using Ratcliff/Obershelp pattern recognition (*R/O*) which evaluates the similarity of the ASR string to a book title from the database (Ratcliff and Metzener 1988). Confidence scores were not displayed on the wizard's GUI.

Given an ASR query, the database produced one of the four following kinds of returns, based on the R/O scores:

- *Singleton*: the single top-scoring candidate, if any were very good (R/O ≥ 0.85)
- *AmbiguousList*: two to five moderately good candidates (0.85 > R/O ≥ 0.55)
- *NoisyList*: six to nine poor but non-random candidates (0.55 > R/O ≥ 0.40)
- *Empty:* No candidates (max R/O < 0.40)

Our focus here is not on the database search, but on the wizard's actions given noisy ASR and an adequate but imperfect database return. Words in each candidate that exactly matched a word in the query appeared darkest on the GUI. All other words appeared in grayscale in proportion to their degree of character overlap with the words in the query.

Two of the seven subjects were non-native speakers of English (one Spanish, one Romanian). Each pair of students (a total of 21 possible pairs) met five times. In each meeting, one student was the user and the other was the wizard in a *session* of 20 title cycles. Then the pair immediately exchanged roles to run a second session of 20 title cycles. Thus, each student was the wizard on 100 title cycles and the user on 100 title cycles with every other student, for a possible 4200 title cycles in all. Users were permitted to end a session early after fewer than 20 title cycles if they experienced severe system problems.

Beyond the mechanics of this process, it was important to create a dialogue-like environment and to encourage the best possible performance from our subjects. To make her speech more conversational and less like simply reading a list, the user prepared immediately before each session. She read brief synopses of the 20 titles (chosen at random from the database) and then ordered them in some way (e.g., genre or theme) relevant to their content. To encourage thoughtful decisions, no time limits were imposed upon either the wizard or the user. Finally, we devised a score that subjects were asked to maximize throughout the experiment, with prizes to be awarded for the top two scorers. The wizard scored +1 for a correctly identified title, +0.5 for a relevant question and -1 for an incorrect title. To encourage cooperation between users and wizards, the user also scored +0.5 for a successfully recognized title.

Results

The analysis in this section provides essential support for automatically learning models of intelligent behavior worthy of incorporation into an SDS. Given the permitted early termination, there were 4172 title cycles (instead of 4200). In them, the average WER was 69%. Nonetheless, the distribution of database returns was 46.7% Singleton, 53.26% AmbiguousList and 2.83% NoisyList. (Although in pilot tests 5% - 10% of the returns were empty, during the experiment itself none were.)

Figure 1 shows the overall distribution of wizard actions for our subjects, W1 through W7. Each of them saw a similar distribution of database returns: Singleton ($\mu = 278.57$, $\sigma = 21.16$), AmbiguousList ($\mu = 300.57$, $\sigma = 16.92$), and NoisyList ($\mu = 16.86$, $\sigma = 4.78$). The correct title was among the candidates returned by the database 71.31% of the time. Singleton returns were the correct title 92.05% of the time. AmbiguousList and NoisyList returns contained the correct title 53.74% of the time.

Ideally, a wizard should identify the correct title if it appears among the candidates, and otherwise ask a thoughtful question that could constructively advance the dialogue. As one might expect from our pilot study, wizards knew what the user meant when they saw it. If the correct title was among the candidates, wizards identified it confidently 68.72% of the time and tentatively 26.53% of the time — 95.25% in all. Recall, however, that AmbiguousLists and NoisyLists were sorted by search confidence. When the database returned multiple candidates, the top candidate was the correct title 41% of the time. It was second 5.81%, third 2.61%, fourth 2.20%, and later (fifth through ninth) 1.67% of the time. This did indeed help the wizards, who correctly offered the first title 98.34% of the time (74.24% confidently, and 24.10% tentatively). Of course, preference for

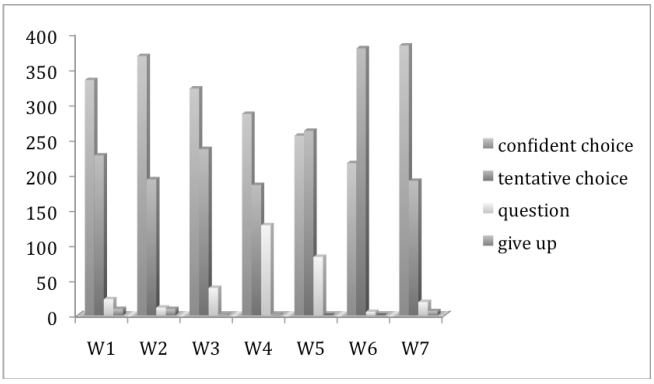

Figure 1: Distribution of wizard actions

the top returned candidate is readily programmed into an SDS. Instead we focus here on what wizards did when the title was not among the candidates.

Wizards were less skilled at recognizing what the user had not meant. Indeed, their performance differed primarily on their response when the correct title was not among the candidates — most wizards were less accurate then, and their performance was less uniform. Despite careful instructions to the subjects that had explained this option, wizards asked a question in only 22.32% of the cases where the correct title was not among the candidates. Instead they made a tentative guess (67.71%), chose confidently (7.78%), or gave up (2.20%). Table 1 shows each wizard's number of title cycles, session score, and *accuracy,* the proportion of title cycles where she identified the correct title or correctly recognized that the title was not among the candidates (by asking a question or giving up). It also shows the frequencies with which she offered the top candidate and correctly recognized that the title was not among the candidates

Wizards are ranked in Table 1 in descending order of session score and accuracy. Those values are highly correlated ($R = 0.91$, $p = 0.0041$). W4 scored highest, primarily because of the frequency with which she asked a question when the candidates did not include the correct title (correct non-offers = 64%). Table 2 shows the distribution of what should have been the correct action across all 4172 title cycles. The correct action was either to offer the title as the correct candidate at a given position (Return 1 through Return 9) or to ask a question or give up when the title was not among the candidates. Table 2 makes clear that the simple strategy "always guess the top candidate" (as our wizards often did) would achieve about 65% accuracy. Note too that those wizards who relied on it most (W3 and W6) were also the least accurate overall, while the wizard who relied on it least (W4) was the most accurate. Clearly, given a reasonable but fallible database search on noisy ASR, an SDS should emulate W4, not simply choose the top candidate.

Learning to be Like a Wizard

Wizards collaborate with the SDS — the system manages input and output (except for the wizard's questions), while the wizard exploits the available information (ASR string and database return) to make a decision. Our experimental design also captured data that described the system and the wizard's session history. That data was then used to train models of wizard actions selection. Such models could be used to implement the best wizard behavior within an SDS.

The experiment collected data on 60 features available at run time, selected for their likely relevance to wizard action choices. They described the ASR (e.g., number of words in the ASR string), the recognition process (e.g., recognizer's confidence score when it produced the ASR string), the speech signal (e.g., speech rate as number of 10ms speech fragments per word), the ability of the SDS to interpret the ASR string (e.g., number of parses in the natural language understanding component), and Olympus/ Ravenclaw confidence scores that combine recognition with language understanding. (Much of this system information was not available to the wizard.) Other features described the session history (e.g., number of correctly identified titles so far), the database return (e.g., return type of Singleton, AmbiguousList, NoisyList), or the similarity between the ASR string and the candidates (e.g., number of matching words). Because the number of candidates differed across title cycles, these features were averaged over multiple candidates.

Table 1: Raw session score, accuracy, proportion of offered titles listed first in the database search return, and frequency of correct non-offers for seven participants.

Subject	Cycles	Session score	Accuracy	Chose #1	Correct non-offer
W4	600	0.7585	0.8550	70%	64%
W5	600	0.7584	0.8133	76%	43%
W7	599	0.6971	0.7346	76%	14%
W1	593	0.6936	0.7319	79%	16%
W2	599	0.6703	0.7212	74%	10%
W3	581	0.6648	0.6954	81%	20%
W6	600	0.6103	0.6950	86%	3%

As a machine learning technique, we chose decision trees to model wizard behavior because they are easy to interpret and compare, and relatively transparent. A decision tree maps feature values to a target value (here, wizard action). A decision tree is a tree-like structure of nodes with directed links between them. Each node is a branch test based on feature values. To simulate the modeled behavior, a program traces a path from the *root* (the top node), following the branch tests until it reaches a *leaf*, a non-branch node that provides a target value. With a version of C4.5 (Quinlan 1993), we trained two kinds of decision-tree models: an *overall model* that used data from all the wizards to predict wizard action in general, and seven individual *wizard models,* one for each wizard.

Cross-correlation over the features indicated that many of the initial 60 features were heavily correlated. We manually isolated groups of correlated features with $R^2 > 0.5$, and retained only one representative feature from each group. We grouped features that described the similarity between ASR string and candidates, features that described the database

Table 2: Distribution of correct wizard actions

Correct action	N	%
Return 1	2722	65.2445
Return 2	126	3.0201
Return 3	56	1.3423
Return 4	46	1.1026
Return 5	26	0.6232
Return 6	0	0.0000
Return 7	7	0.1678
Return 8	1	0.0002
Return 9	2	0.0005
Question | give up	1186	0.2843
Total	4172	1.0000

search returns, features that described confidence scores from various system components, and features that described the speech signal. This left 28 features. Before training each model we also ran CfsSubsetEval, an attribute selection algorithm that evaluates subsets of features based on both their individual predictive power and the degree of redundancy among them (Hall 1999). This further reduced the number of features to between 8 and 12 per model. (Many of the same features survived into more than one model.) To reduce overfitting, we also activated pruning to remove subtrees likely to provide little additional power because they cover too few training instances.

To confirm the learnability and quality of the decision trees, we also trained logistic regression and linear regression models on the same data. Here, regression captures the change in wizard action based on the changes in feature values (Witten and Frank 2005). Linear regression fits data to a linear function, and represents the wizard's four actions numerically in decreasing value: confident choice, tentative choice, question, and give up. Logistic regression predicts the probability of an action based on fit to a logistic curve. This generalizes the linear model to predict categorical data, here, the wizard's four actions. All models were produced with the Weka data mining package (Hall, et al. 2009) under 10-fold cross-validation.

Ability to predict wizard action was uniform across learning methods. On the overall model, logistic regression had 75.2% accuracy while the decision tree's accuracy was 82.2%. The linear regression model had root mean squared error of 0.483, while the decision trees' was 0.306. Predictive ability for the individual wizard models was similarly comparable. Thus the remainder of this discussion is restricted to decision trees.

Table 3 describes the learned models for individual wizards (ranked by wizard accuracy from Table 1). It shows size in number of nodes, number of included features, accuracy, and the F measure on confident choice. Note that model accuracy does not correlate with wizard rank; model accuracy indicates only how well the tree predicts the wizard's action from the training data. The simplest wizard strategies (e.g., always select the top candidate) are clearly easier to predict, but not necessarily better. (Compare, for example, W4 and W6.)

Recall from Figure 1 that confident choice was more common than tentative choice, which was in turn more common than question or give up. As a result, the individual models consistently predicted a confident choice with $0.80 \leq F \leq 0.87$, but less consistently predicted tentative choices with $0.60 \leq F \leq 0.89$, and could predict question only for W4, the top-scoring wizard who most often asked questions.

Table 3: Learned decision trees model individual wizards.

Tree	Rank	Size	Features	Accuracy	F conf
W4	1	55	12	75.67	0.85
W5	2	21	10	76.17	0.85
W1	3	7	8	80.44	0.87
W7	4	45	11	73.62	0.83
W3	5	33	10	77.42	0.84
W2	6	35	10	78.49	0.85
W6	7	23	10	85.19	0.80

The features that appeared most often in the individual models primarily described the database return, the ASR string's similarity to the candidates, the wizard's recent performance, and the quality of the speech recognition and language understanding. (Note that the last two were not available to the wizard.) The five features that appeared most often at the root or top-level nodes were

- *ReturnType* (Singleton, AmbiguousList, NoisyList)
- *RecentSuccess*, how often the wizard had chosen the correct title within the last three title cycles
- *ContiguousWordMatch*, the maximum number of contiguous word matches between a candidate and the ASR string (averaged across candidates)
- *NumberOfCandidates*, how many candidates were returned by the database
- *Confidence*, an Olympus/Ravenclaw metric on confidence for recognition and language understanding

Careful inspection of the model for the most accurate wizard (W4) indicates that, if ReturnType was NoisyList, she asked a question. If ReturnType was AmbiguousList, her decision involved the five features above, plus the acoustic model score (another internal system measure that indicates the quality of the speech recognition), the length of the ASR string in words, the number of times the wizard asked the user to repeat, and the maximum size of the gap between matching words in the ASR string and the candidates. To further focus our analysis on W4's distinctive behavior, we trained an additional decision tree to model how W4 chose between selecting a title and asking a question. The resulting model on 600 data points (each corresponds to a title cycle) consisted of 37 nodes and 8 features, with F = .91 for selecting a title and F = 0.68 for asking a question. The root of this tree differs from all other wizard models — it is the number of frames (10ms speech segments used to produce the ASR string), a measure of the length of the ASR. On short ASR strings (as measured both in number of frames and number of words) with AmbiguousList or NoisyList returns, W4 asked a question when RecentSuccess $\leq$ 1 or ContiguousWordMatch = 0, and the acoustic model score was low. (Short titles are more readily confused.) On long ASR strings, W4 asked a question only when ContiguousWordMatch $\leq$ 1, RecentSuccess $\leq$ 2, and either the return was a NoisyList, or Confidence was low and there was more than one candidate. In summary, the factors that drove W4 to ask a question include the length of the ASR string, the quality of the ASR transcription, the database return type, the similarity between the ASR string and the candidates, and how well she had performed on recent title cycles. These can all be captured by system-internal features.

Discussion and Future Work

As used here, wizard ablation embeds a wizard within an SDS to study her choices when placed in the same environment as a machine. Given noisy ASR and the results of a

database search, the best wizards do not always guess based on search return. Instead they sense that the knowledge they have is a poor fit with what the recognizer "heard." In that case, a good wizard infers that the correct title is not among the returned candidates, and asks a thoughtful question to move the dialogue forward. (The mystery book at the beginning of this paper, by the way, was the third title listed.)

Experiments like this provide insight into how people match noisy input with returns from database search. The experimental design led wizards to prefer the first candidate listed — they read it first, and it was typically correct if the return included the correct title. Thus a wizard's skill at finding the title when it is present is less noteworthy than W4's ability to question the relevance of all the candidates.

The focus here has been on a single book request by title. Current work extends this approach to full dialogue. Wizards will see ASR and query results, and will have a predefined set of system-actions from which to choose. Dialogue interactions will include greeting, user identification, and four book requests by author and catalogue number, as well as by title. In full dialogue, context will have more relevance and can be measured more realistically by metrics in addition to RecentSuccess, Analysis of wizards' questions from this experiment will motivate a pre-defined set of questions for wizards in the full dialogue study.

This work successfully learned models that predict wizard action primarily from system features. (The only prevalent wizard-specific feature was RecentSuccess, which is readily replaced by the system's recent success.) Similar learned models will be incorporated into CheckItOut. Our next experiment will train models to predict wizards' actions during full dialogue with our baseline version of CheckItOut, and then refine the system with the learned models. We predict that evaluation of the refined, wizard-informed CheckItOut will provide better performance.

Acknowledgments

This research was supported in part by the National Science Foundation under IIS-084966, IIS-0745369, and IIS-0744904. We thank the staff of the Heiskell Library, the Olympus/Ravenclaw developers at Carnegie Mellon, and our tireless undergraduate research assistants.

References

Aist, G. S., Allen, J., Campana, E., Gomez Gallo, C., Stoness, S., Swift, M. and Tanenhaus, M. K. (2007). Incremental dialogue system faster than and preferred to its nonincremental counterpart. *CogSci 2007,* 779-774. Nashville, Tennessee.

Bohus, D. (2004). *Error Awareness and Recovery in Task-Oriented Spoken Dialog Systems. Ph.D. Thesis*. Computer Science Carnegie Mellon University.

Bohus, D., Raux, A., Harris, T. K., Eskenazi, M. and Rudniky, A. I. (2007). Olympus: an open-source framework for conversational spoken language interface research. *Proceedings of Bridging the Gap: Academic and Industrial Research in Dialog Technology workshop at HLT/NAACL 2007,* 32-39.

Bohus, D. and Rudniky, A. I. (2003). RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. *Proceedings of Eurospeech 2003,* 597-600.

Dix, A., Finlay, J., Abowd, G. D. and Beale, R. (2003). *Human-Computer Interaction*, Prentice Hall.

Forbes-Riley, K. and Litman, D. (2009). Adapting to Student Uncertainty Improves Tutoring Dialogues. *Proceedings of the 14th International Conference on Artificial Intelligence in Education, AIED,* 33-40. Brighton, UK.

Franceschetti, D. R., Adcock, A. B. and Graesser, A. C. (2003). Analysis of strategies in expert tutoring dialog for use in Intelligent Tutoring System Development. *CogSci 2003,* 1344. Boston, Massachusetts.

Hall, M. (1999). *Correlation-based Feature Selection for Machine Learning*. *Ph.D. Thesis*. Department of Computer Science University of Waikato.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations* 11(1).

Levin, E. and Passonneau, R. (2006). A WOZ variant with contrastive conditions. *Proceedings of the Interspeech Satellite Workshop, Dialogue on Dialogues: Multidisciplinary Evaluation of Speech-Based Interactive Systems,* 17-21.

Ohlsson, S., Corrigan-Halpern, A., Di Eugenio, B., Lu, X. and Glass, M. (2003). Explanatory Content and Multi-Turn Dialogues in Tutoring. *CogSci 2003,* 48. Boston, Massachusetts.

Passonneau, R., Epstein, S. L. and Gordon, J. B. (2009). Help Me Understand You: Addressing the Speech Recognition Bottleneck. *AAAI Spring Symposium on Agents that Learn from Human Teachers,* 119-126. Paolo Alto, CA.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Ratcliff, J. W. and Metzener, D. (1988). Pattern Matching: The Gestalt Approach. *Dr. Dobb's Journal* 7, 46.

Raux, A. and Eskenazi, M. (2004). Non-Native Users in the Let's Go!! Spoken Dialogue Systems: Dealing with Linguistic Mismatch. *HLT/NAACL,* 217-224. Boston, MA.

Rieser, V. and Lemon, O. (2006). Using Machine Learning to Explore Human Multimodal Clarification Strategies. *COLING/ACL-06,* 659-666. Sidney, Australia.

Skantze, G. (2005). Exploring human error recovery strategies: Implications for spoken dialog systems. *Speech Communication* 45(3), 325-341.

Stoyanchev, S. and Stent, A. (2009). Predicting Concept Types in User Corrections in Dialog. *EACL Workshop SRSL,* 42-49.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, Morgan Kaufmann.