Data Mining to Support Human-Machine Dialogue for Autonomous Agents

Susan L. Epstein[1], Rebecca Passonneau[2], Tiziana Ligorio[1], Joshua Gordon[3]

[1] Hunter College and The Graduate Center of The City University of New York, Department of Computer Science, New York, NY
[2] Center for Computational Learning Systems, Columbia University, New York, NY
[3] Department of Computer Science, Columbia University, New York, NY
{susan.epstein@hunter.cuny.edu, becky@cs.columbia.edu, tligorio@gc.cuny.edu, joshua@cs.columbia.edu}

Abstract. Next-generation autonomous agents will be expected to converse with people to achieve their mutual goals. Human-machine dialogue, however, is challenged by noisy acoustic data, and by people's preference for more natural interaction. This paper describes an ambitious project that embeds human subjects in a spoken dialogue system. It collects a rich and novel data set, including spoken dialogue, human behavior, and system features. During data collection, subjects were restricted to the same databases, action choices, and noisy automated speech recognition output as a spoken dialogue system. This paper mines that data to learn how people manage the problems that arise during dialogue under such restrictions. Two different approaches to successful, goal-directed dialogue are identified this way, from which supervised learning can predict appropriate dialogue choices. The resultant models can then be incorporated into an autonomous agent that seeks to assist its user.

Keywords: spoken dialogue systems; Wizard of Oz; human-machine interaction.

1 Introduction

A *spoken dialogue system* (*SDS*) is an autonomous agent that communicates with people in the way most natural to them — through spoken language. In pursuit of a common goal, however, such an agent must not only speak to the person, but also listen. People want human-machine dialogue to be both *successful* (accomplish their goal) and *habitable* (demonstrate people's tacit knowledge about how dialogue should be conducted). To that end, this paper studies how one person manages to help another achieve a goal during interactions that simulate dialogue between a human and an autonomous agent. The primary result reported here is the identification of two different strategies for effective, goal-directed dialogues. One is service oriented, and

the other is data driven. Each could serve as a model for an autonomous agent engaged in goal-directed human-machine dialogue.

This paper describes the collection and mining of a rich corpus in an ambitious domain. The corpus describes dialogues between two people: a person (here, the *caller*) and a *wizard*, whom the caller believes to be a computer system but is actually another human. The wizard, however, is *ablated*, that is, her input and permitted actions are restricted to those that would be available to an autonomous agent [1]. She is also *embedded* [2], that is, the wizard and the system process some of the same inputs concurrently, so that run-time system features can be mined to model the wizard's actions. Thus the wizard experiences dialogue much the same way that an SDS would — she accepts automated transcriptions of human speech as input and generates speech as output.

We emphasize that the wizard is not intended to be a model of an expert system, but rather a model of how to solve problems that arise during dialogue. Therefore the wizard need not be quick or even perfectly accurate. Rather, the wizard should try to understand what the caller requires from what the caller says, and help achieve that goal. Because automated speech recognition for arbitrary speakers across a large vocabulary is extremely difficult, however, the wizard must actually try to understand what the caller requires from a noisy transcription of what the caller has said.

The assembled corpus includes caller speech, wizard behavior, and a broad spectrum of descriptive features available to a traditional SDS. From it, we identify the most expert wizards, and examine how they made their decisions. The next section of this paper provides details on our domain of investigation. Subsequent sections discuss related work, and summarize the preliminary experiments that supported the approach in a full dialogue experiment. The paper then details the design and results of that experiment, and discusses them and current work.

2 Speaking with the Librarian

Our real-world domain presents considerable challenges, but also provides an extensive supporting knowledge base. We investigate book requests to the Andrew Heiskell Braille and Talking Book Library, a branch of the New York Public Library and a Regional Library of the National Library Service for the Blind and Physically Handicapped of the Library of Congress. Most of Heiskell's holdings are books recorded in a proprietary format and mailed to its patrons on request. Heiskell's patrons receive a monthly newsletter that describes the newest titles, along with their authors and catalog numbers.

Patrons order books from a Heiskell librarian by telephone. As they speak with the patron, Heiskell's librarians search their *book database* of titles and authors, to identify what they believe the patron wants. As of 2007, there were only 5 librarians to serve 5,028 active patrons, many of whom order books several times each week. At that time, there were no plans to increase the staff, and even the recording mechanism for off-hours calls was at capacity. Clearly an autonomous agent that could handle the simplest of these calls would be of great value, both to Heiskell's patrons and to those of other libraries that provide a similar service.

CheckItOut is an SDS that accepts mock "book orders" from callers on its dedicated *VOIP* (Voice Over Internet Protocol) line. During a call, CheckItOut introduces itself; identifies the caller; handles up to four book requests by title, author, or catalog number; summarizes the order if the caller wishes; and signs off. CheckItOut has available to it the 2007 versions of Heiskell's full book and transaction databases, and a sanitized version of Heiskell's database on its active patrons. Only the catalog number is guaranteed to identify a book uniquely.

An SDS receives a continuous stream of acoustic data that includes background noise as well as human speech. The SDS relies on *ASR* (Automated Speech Recognition) to translate that input into one or more *recognition hypotheses* (sequences of words, or text strings). CheckItOut's speech recognition is intentionally not state-of-the-art, because we seek to develop interpretation methods and dialogue strategies that are robust to noisy ASR. During dialogue with a caller, CheckItOut must contend with a large vocabulary (54,448 distinct words alone from 71,166 titles and 28,031 authors), a varied speaker population, and callers in environments with background noise, all of which make ASR considerably more difficult. Table 1 provides some examples of noise-ridden ASR output for spoken book titles.

Table 1. Book titles and their noisy ASR output.

	Title	ASR output
1	*Into the Night*	Into than 9
2	*Helen and Teacher: The Story of Helen Keller and Anne Sullivan Macy*	Helen an teacher distort tell until an am Sullivan Macy
3	*Map of Bones*	Nah don't bones
4	*I Lived to Tell it All*	Elusive total man

Clearly, noisy ASR is likely to produce both misunderstandings (incorrect semantic interpretations) and *non-understandings*, where the system cannot interpret the ASR at all. There is signal within the noise, however, and our previous work shows that people identify it well, given sufficient context. We seek to develop an SDS that does so too.

3 Related Work

3.1 Spoken Dialogue Systems

CheckItOut is built within a traditional *pipeline architecture* for SDSs, shown in Figure 1. In this architecture, an *audio manager* analyzes the incoming acoustic stream for voice activity, and forwards a sequence of overlapping audio frames to an *interaction manager*. The interaction manager (*IM*) performs an initial segmentation (determines utterance boundaries) that it sends to the ASR module. The ASR module forwards its recognition hypotheses (as orthographic text strings) to a natural language understanding (*NLU*) module. The NLU produces one or more parses, each

of which is a mapping from ASR to concepts, and then a confidence annotator identifies the best parse.

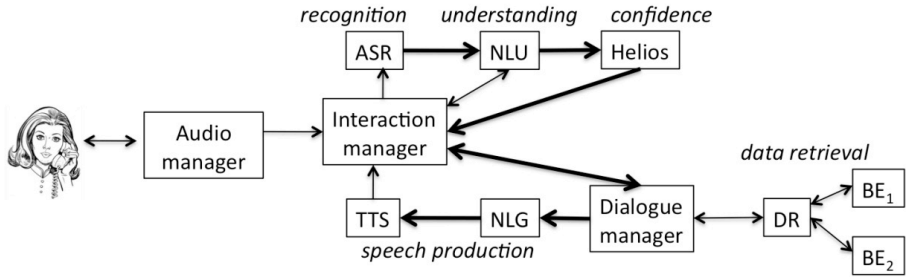

Fig. 1. A traditional pipeline architecture for spoken dialogue systems. Heavy arrows emphasize the pipeline.

The focus of our work here is the decision maker: the dialogue manager and its relationship to the four modules (IM, ASR, NLU, and confidence annotator) involved in spoken language understanding. (Ultimately, the IM uses input from the NLU, the confidence annotator, and the dialogue manager to decide where the next utterance boundary lies, and thereby to which parsed concepts the dialogue manager should respond.) The dialogue manager decides what action to take next. It can use the user-designed, application-specific *domain reasoner* (*DR*) to query the knowledge stored in one or more backends (*BE*), or it can decide to speak. A decision to speak is forwarded to the natural language generator (*NLG*), which formulates text and then passes it to a text-to-speech (*TTS*) module. Finally the speech output is directed to the IM, which transmits it to the caller. Each such utterance is referred to here as a *system prompt*.

The *Olympus/RavenClaw* pipeline architecture [3, 4] has been the framework for at least a dozen SDSs. The components in an Olympus/Ravenclaw SDS communicate through the Galaxy hub [5]. Olympus/Ravenclaw supports a variety of modules for each of its components. CheckItOut, our SDS for the simulated Heiskell library world, uses the *Apollo* interaction manager [6, 7] and the *PocketSphinx* speech recognizer [8]. For our domain we adapted freely available acoustic models of Wall Street Journal dictation speech with about eight hours of spontaneous speech. CheckItOut also uses the *Phoenix* context-free grammar semantic parser [9], the *Helios* utterance-level confidence annotator [10], and the *Kalliope/Swift* TTS [11]. CheckItOut's dialogue manager is built within *RavenClaw*, which separately defines domain-dependent conversational goals in an explicit task hierarchy and provides domain-independent error-handling strategies.

3.2 Challenges in Spoken Dialogue

In commercial SDSs intended for many callers, accurate speech recognition performance is achieved at the price of a system that generates an inflexible sequence of prompts, has limited strategies to address communication difficulties, and handles

very small subsets of language. These factors are partially responsible for the frustrating telephone conversations people often experience with them. Speech recognizers rely on pre-existing acoustic models to relate acoustic energy to speech sounds, and on trained, domain-specific language models to predict which sequences of speech sounds correspond to known words. Although large-vocabulary ASR has improved dramatically for single-party applications, such as the transcription of broadcast news, its accuracy in dialogue lags substantially.

To limit communication errors incurred by faulty ASR, an SDS may use enriched strategies to detect and respond to incorrect recognition output [12]. It may repeatedly request caller confirmation to avoid misunderstanding. This confirmation may be either *explicit* (e.g., "I heard you say *Grapes of Wrath*. Is that correct?") or *implicit* (e.g., "By Steinbeck. That's available."). Implicit confirmation uses language that elicits responses from the caller that the system can handle [6]. If the caller adds new information in response to a system prompt, two-pass recognition can consider the extra information contained in that response to restrict the language expected in the second pass and thereby achieve better recognition [13].

Despite careful engineering in the laboratory, ASR performance in fielded research dialogue systems commonly has a word error rate (*WER*) at best near 30%-35% and as high as 70% [4]. One way to minimize both misunderstandings and non-understandings despite high WER is with accurate semantic interpretation. The need to correct the system's misunderstandings, however, can frustrate the caller, and can elicit speech that is more poorly recognized than non-correction utterances [14]. When an SDS re-prompts the caller for the same information after non-understanding, it often fails to understand because the caller then hyperarticulates, which generally results in even worse recognition.

Another source of caller frustration during human-computer dialogue is that the SDS maintains *system initiative*, that is, it alone controls the path of the dialogue. This results in a rigid sequence of spoken commands to the caller. In contrast, a *mixed-initiative* SDS permits the caller to volunteer information, to interrupt or to correct the system when it fails to understand the caller's intent. Once a system shares the initiative with the caller, however, it risks confusion. RavenClaw's dialogue task tree offers a way for the SDS to remember and anticipate what is under discussion. Thus, if a CheckItOut caller signals that the system has misunderstood her with "That's not what I said," CheckItOut will remove the incorrect book from the order. The system also uses a subset of RavenClaw's domain-independent error-handling strategies. Among these is one that, after three or four consecutive non-understandings, has the SDS abandon the call, that is, simply hang up.

3.3 Wizardry in Human-machine Dialogue

During dialogue, people may fail to understand or may misunderstand one another. To avoid frustration, people often use creative ways to re-elicit needed information — they use contextual clues to fill in gaps, and to confirm that some communication has occurred. We seek to exploit for an autonomous agent a similar repertoire of responses to support habitable and successful dialogue.

To acquire knowledge about human strategies that would support robust interpretation of noisy ASR, we use a *Wizard of Oz* study, where, unbeknownst to the human caller, another person (the wizard) plays the role of the system. Wizard of Oz studies were originally intended to elicit caller behavior, so that system designers could anticipate likely system input [15]. They have also been used to study how people make decisions when given the same input and set of actions that would be available to a system. Figure 2 portrays how a wizard can be embedded in an SDS, with the wizard's dataflow indicated by heavy arrows.

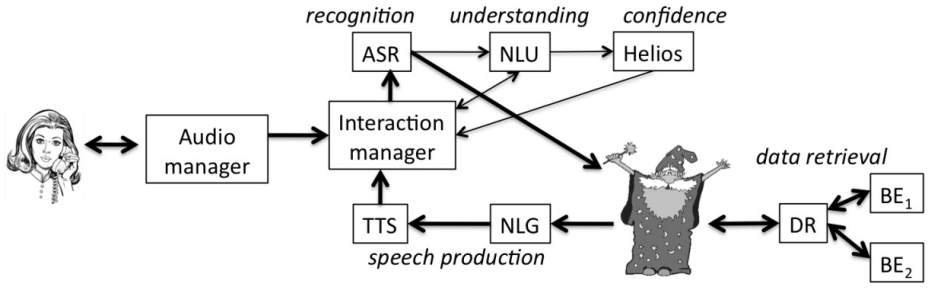

Fig. 2. A wizard embedded in CheckItOut. Heavy arrows indicate the dataflow for the wizard.

Other Wizard of Oz studies that focused on the wizard during full spoken dialogues have included efforts to predict the wizard's response when the caller is not understood [12], the wizard's use of multimodal clarification strategies [16], and the wizard's use of application-specific clarification strategies [17]. The full dialogue experiment presented in Section 5 differs from that work in three ways (detailed in our earlier work on partial dialogue [2]): it analyzes several wizards' behavior, it recognizes differences among wizards, and it identifies distinctive and successful behavior, so that the SDS will ultimately benefit only from models of the most skilled wizards.

Wizards interpret noisy ASR much better than machines do, because they know which aspects of context are relevant. Given real or simulated ASR output instead of the caller's speech during dialogue, human subjects engage in problem solving about the task, the ASR errors, or both [18]. One particularly useful technique is *voice search*, where the wizard directly queries a knowledge base with ASR output, and receives returns ranked by a similarity score [19]. Voice search provides context to seemingly unintelligible ASR output. For example, SOONER SHEEP MOST DIE is readily resolved in the context of the candidate book title matches *Soon She Must Die*, *Why Someone Had to Die*, and *The Messenger Must Die* [2].

4 Preparatory Experiments

This section summarizes several experiments, first reported elsewhere, that were performed in preparation for the full dialogue experiment in Section 5. They demonstrated that CheckItOut's task is more difficult than that of a well-known,

fielded system, and that people acting as ablated wizards are surprisingly good at CheckItOut's book order task.

4.1 Task Exploration

Caller requests for books by title are more challenging than those by catalog number or author, because the book title field is much more like free text than like structured data, and thus much less predictable. Under four WERs (from perfect to very poor), we compared how difficult it was for the Phoenix parser to identify the request type of callers' utterances in two Olympus/RavenClaw systems [20]. We parsed requests for books by title and by author from CheckItOut, and requests for bus information by route and by destination place name from Let's Go Public! [4]. Caller utterances in CheckItOut proved longer and more challenging at every level of WER. For example, at the same WER of 0.4, concept accuracy was about 74% for Let's Go Public! but only about 36% for CheckItOut. Thus the semantic parsing approach used by CheckItOut and Let's Go Public! has more difficulty understanding how to interpret input to CheckItOut as a request type. In contrast, an alternative machine-learning approach to NLU performed much better on the longer utterances in CheckItOut (85%) than the short utterances of Let's Go Public! (36%).

In an offline *pilot experiment* of requests for books by title, we also demonstrated that people were remarkably effective at matching error-ridden ASR strings to items in a large text file. We trained the language model for PocketSphinx on a random sample of 500 Heiskell book titles, which contained 1,400 distinct words. Each of the three participants then received, in text format, CheckItOut's ASR output from a single speaker on 50 randomly-chosen titles from those 500. (ASR output quality was even poorer than in Table 1. WER ranged from 0.69 to 0.83, depending upon the speaker.) Participants were given the frequency with which each individual word occurred in the 500 titles, the borrowing frequency for each book, and a text file of all 71,166 book titles. Participants were asked to identify or guess the title in each instance, and were permitted to search the text file in any way they chose, with no time limit. Although the ASR rendered only 9% of all 150 titles perfectly, participants identified a startling 61.7 – 71.7% of their books correctly. Subjects were also asked to explain how they made their decisions. See [21] for further details.

4.2 The Book Title Experiment

People's ability to make sense of poor ASR output in the pilot experiment motivated a large-scale Wizard of Oz *book title experiment*. In our domain, requests for books by title pose the greatest challenge. Thus our initial ablated wizard study had multiple wizards of varying expertise address partial dialogues consisting of a single *turn exchange*: the caller spoke a title and in response the wizard either offered a book title or asked a question. With 7 subjects who played both roles, we collected 4,172 book title requests with poor ASR between all possible caller-wizard pairs.

The caller and the wizard each had a graphical user interface (*GUI*) and a microphone. When the caller read a book title into a speech recognizer through her

microphone, the corresponding ASR appeared on the wizard's GUI, where the wizard formulated a query from the ASR for the database. Given the results of the query, the wizard then indicated on her GUI what she believed to be the correct book, asked through her microphone a question that she believed would help identify the correct book, or indicated on her GUI that she gave up. The caller scored any wizard-identified title as correct or incorrect, and that score was displayed on the wizard's GUI. (The caller also scored any questions, but that information was not shared with the wizard.) All 7 wizards could identify correct matches returned by their query, with individual accuracy that ranged from 69.5% to 85.5%. For further details on the mechanics of this process, how we created a dialogue-like environment, and how we encouraged the best possible performance from our subjects, see [2, 22, 23].

During this experiment, we collected data on 60 features available at run time. These features were motivated by comments from the subjects in the pilot experiment on how they matched titles to the database; we identified those likely relevant to dialogue management. System features, most of which were unavailable to the wizard, described the speech signal, the ASR output, the recognition process, the ability of the SDS to interpret the ASR string, and two Olympus/Ravenclaw confidence scores that combine recognition with language understanding. Other features described the session history (e.g., number of correctly identified titles so far), the database return (e.g., number of returned titles), and similarities between the ASR string and the returned titles (e.g., number of matching words).

Although wizards regularly detected correct matches, few responded appropriately when the correct title was not returned by the query. In that case (28.43% of all turn exchanges), the wizard should have asked a question. Despite careful instructions to the contrary, all wizards did so in only 22.32% of those situations. Only the two most accurate (85.50% and 81.33%) wizards recognized when none of the database returns was a likely match; they asked questions in 64% and 43% of these situations, respectively. The next most frequent questioner asked only 20% of the time. Clearly, recognition that no match is present is important to accuracy in this task.

We used linear regression, logistic regression, and decision trees on the features monitored during the book title experiment to model the behavior of each individual wizard and of the wizards as a group. For the all-wizard models, logistic regression had 75.2% accuracy, and decision trees 82.2%. Linear regression had root mean squared error 0.483, and decision trees had 0.306. The predictive ability of the models for individual wizards was similar. The features our best wizards used, their propensity to ask questions, and the kinds of questions they asked provided further guidance for the full dialogue experiment recounted in Section 5.

4.3 A Baseline for the Wizard Dialogues

To establish a baseline in CheckItOut for comparison of human-wizard dialogues with human-system dialogues, 562 dialogues were collected, with 10 caller subjects, 5 male and 5 female. Each caller made 50 book-order calls to CheckItOut over three days. For each call, the subject received (from a web site) a new *scenario*: a patron identity, a list of four books, and instructions to request, in any order, one book by catalog number, one by title, one by author, and one by any of those methods.

CheckItOut's domain reasoner performs a partial matching against the database on the words in the parse in which Helios had the most confidence. This procedure uses Ratcliff/Obershelp pattern recognition (*R/O*) [24] to evaluate the similarity of a submitted string to a book title, author, or catalog number in the database. The *R/O score* is the number of matching characters divided by the total number of characters. For example, for ROLL DWELL the three top-candidate titles and their R/O scores were *Cromwell* (0.666), *Colin Powell* (0.636), and *Robert Lowell* (0.608). On average during the baseline experiment CheckItOut identified 2.39 books correctly on each call (range 0 – 4, standard deviation $\sigma = 1.3$).

5 Experimental Design

The *full dialogue experiment* described in this section serves as a paradigm for the development of dialogue skills for autonomous agents. It places wizards in a more challenging environment, one almost identical to that of an autonomous dialogue agent, except that instead of responding promptly to callers' turns, wizards could take time to problem solve. Voice transmission was by telephone, and book requests were by title, author, or catalog number. Caller and system participated in a dialogue, complete with *disfluencies* (e.g., pauses, repetitions, and self-corrections), corrections ("That's not what I said"), and subtasks (e.g., identification of the caller, task summary). People were thereby freed to pursue the same conversational goals by creatively different means.

5.1 Subjects and Preparation

Both wizards and callers were recruited by email and flyers to students at three local universities. We solicited volunteers for the (more difficult and more remunerative) role of wizard. A single trainer instructed four male and five female wizard trainees, one at a time. To familiarize them with the custom database query (described in Section 4.2), trainees were given 24 ASR strings with 5 candidate search results from the book title experiment, and asked to select which, if any, of the search results matched the ASR. Then each trainee was given a visual and verbal description of a new wizard GUI. (The wizard's GUI in the book title experiment offered fewer choices and less information. The new one is described in Section 5.2 below.) The trainee watched the trainer perform as wizard on a sample call. Finally, each trainee made five test calls during which she could ask questions and talk to the trainer. We then chose as wizards those trainees who were most motivated and skilled at the task. Trainees who were not selected as wizards did not participate further in the experiment.

Callers assumed that they were speaking to an SDS. The lack of real-time response was explained by telling callers that the system was highly experimental, and would be developing and rejecting many hypotheses before responding. They were forewarned that the calls would be long and frustrating. Although wizards were instructed to wrap calls up after six minutes, callers were not permitted to terminate

any call. Each caller also made five training calls, during which she could question the trainer by chat.

5.2 Software

Phoenix is a semantic parser that typically does not rely on syntactic structure, a feature that makes it more robust to ASR noise. A Phoenix parse is a semantic frame consisting of a set of concepts; each concept has its own context-free-grammar (*CFG*). For example, a phone number parse has an area code concept and a concept for the remainder of the phone number. Noisy tokens in an ASR string can be skipped between frames or between concepts, and wild cards can be used sparingly to handle noise within concepts. Such a grammar can handle concepts like phone numbers that have a small vocabulary (digits) and a fixed length, but do not extend well to concepts like book titles with large vocabularies and varying length. To produce rules for the Phoenix book title CFG that preserved its robustness to noise while adding syntactic information, we wrote a transducer to produce Phoenix rules from MICA parses of book titles. (MICA is a broad-coverage dependency parser [25].) Before this experiment, we produced Phoenix rules this way for 3,000 books randomly selected from the 71,166 in Heiskell's database. We then constructed a web page that, on demand, randomly generated a scenario with patron identity (telephone number, name, and address) plus a list of four books randomly selected from those 3,000 titles. As in the baseline experiment, the scenario provided the title, author, and catalog number for each of the four books.

As in Figure 2, the *wizard version* of CheckItOut that answered the telephone included all the modules in Figure 1 and the customized query and matching facilities described in Sections 4.2 and 4.3. In the wizard version, however, the wizard replaced the dialogue manager. Although the Phoenix parser and the Helios confidence annotator did not participate in decisions or select ASR strings for queries, features that described their output and performance were captured for data mining.

A wizard interacted with a caller through two similarly-organized GUIs, one for the login and the other (shown in Figure 3) for the book requests. (This paper focuses on the latter, a considerably more complex task.) In Figure 3, the two frames at the top contain scrollable output from the ASR for the entire dialogue, and from the book database. The material in the two center frames provides the wizard with a dialogue history and sets of basic and auxiliary actions (described further in Section 5.4). The dialogue-history frame in Figure 3 displays how many books have been ordered in the call thus far, their titles, how many questions the wizard has asked, and how often she has asked the caller to repeat. The four frames at the bottom offer *clarifications,* prompts intended to advance the dialogue when the wizard cannot formulate a query or cannot match a book to the current ASR.

The clock in the upper left in Figure 3 changed color after 6 minutes. Wizards were instructed to complete the current book request at that point if it were almost identified, and then end the call, even if all four books had not yet been ordered. This reflects both our desire to minimize the caller's frustration, and our focus on problem solving behavior rather than full task completion.

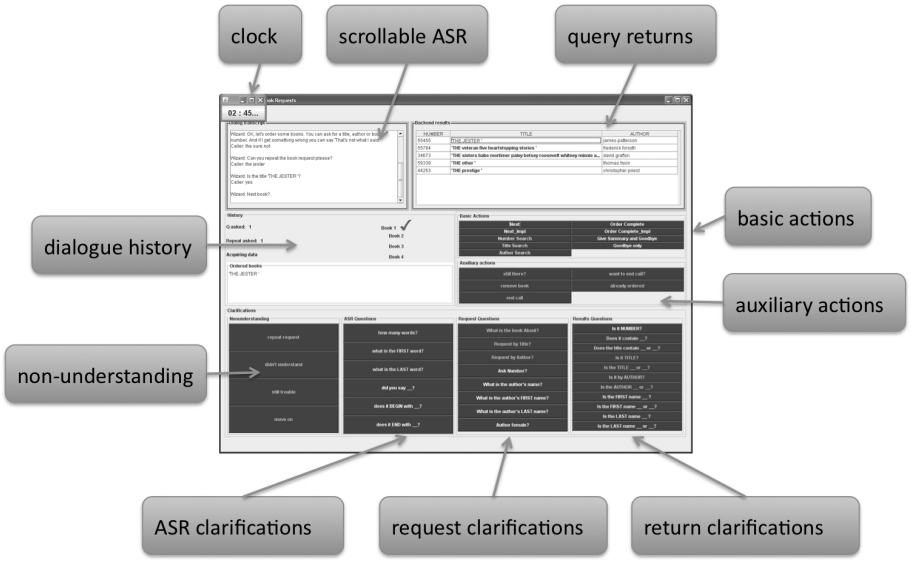

Fig. 3. Annotated screen shot of the wizard GUI during dialogue.

5.3 The Task

Before each call, the caller accessed the scenario web page for an identity and a list of books. The caller was instructed to first identify herself during the *login* for patron identification, and then *request* (orally ask for) the four books in any order: one by title, one by author, one by catalog number, and the fourth by any of those request types. In the full book database, 35% of the titles contain a *subtitle* (an extra phrase that follows the title and is separated from it by a colon). In the random sample of titles covered by the scenarios for this experiment, 39% contained subtitles. On any given title request, callers spoke the title, the subtitle, or both.

Callers used a telephone of their choice (either cellular phone or landline) to call the wizard version of CheckItOut, which answered each call with its customary greeting. Then the wizard took over. The wizard responded only by selection of a pre-specified action on the GUI. Any action that involved speech to the user was transmitted to the NLG. Unless awaiting the caller's response to a system prompt, the wizard was expected to select some sequence of actions and clarifications to advance the dialogue.

5.4 How Wizards Conducted Dialogues

The basic actions on the GUI are either database queries or prompts that the wizard can use to advance the dialogue. Figure 4, drawn from the corpus, shows how a proficient wizard used clarifications to decipher error-ridden ASR. For reference here,

lines are numbered, and *W* denotes speech generated when the wizard clicked a selection on the GUI.

Basic actions on the GUI allow the wizard to request the next book (with or without implicit confirmation of the book just ordered), inform the caller that the order is complete (explicitly or implicitly), offer the caller an optional summary of the order, or say goodbye. Line 1 in Figure 4, for example, is a basic action used by the wizard to obtain a new book request from the caller. C denotes ASR output based on speech from the caller. Although W lines and C lines for the most part alternate, as one would expect in a dialogue, note that lines 6 and 7 represent ASR output for what the system perceived as two consecutive utterances from the caller.

1 W: Next book?
2 C: .PAGAN. .GEORGE. AMERICA
3 W: Did you ask for a title?
4 C: YES
5 W: Can you repeat the book request please?
6 C: .PAGAN. .TURNER. DIABETES
7 C: DIDN'T I AM DONE
8 W: Did you speak the word "diabetes"?
9 C: YES
10 W: Is the title *Conquering Diabetes: A Cutting Edge Comprehensive Program for Prevention and Treatment*?
11 C: NO
12 W: What is the author's last name?
13 C: .STARDANCE. SERVICES
14 W: Is the title *Controle Su Diabetes Guia Para El Ciudado De Su Salud: Take Charge Of Your Diabetes A Guide For Care*'?
15 C: YES
16 W: OK, we now have four books. Would you like a summary of your order?

Fig. 4. Part of a wizard dialogue for one book request. Italics provided for clarity. All caller (C) language is ASR output; all wizard (W) language is generated by the system in response to a command from the wizard.

The book the scenario had assigned to the caller in Figure 4 was *Controle Su Diabetes Guia Para El Ciudado De Su Salud: Take Charge Of Your Diabetes A Guide For Care* whose author the database records as "US Dept of Health and Human Services." The ASR suggests that the caller ignored the Spanish title and read only the English. When a word is bracketed by periods (e.g., .PAGAN. in line 2) it denotes a low confidence score from the ASR. Line 2 offered little information, but the wizard suspected it was a title, which the caller confirmed, so the wizard asked for the title again (lines 3-5). The quality of the ASR output did not improve, but this time it included an unusual and confident word (DIABETES).

Some clarifications indicate to the caller that the system has not understood the last ASR output (request to repeat, explicit statement of non-understanding, repeated statement of non-understanding, decision to go on to the next book). Others ask about what the wizard saw in the ASR (e.g., "How many words?"), about words in the ASR (e.g., "Did you say __?"), about words in the search returns (e.g., "Does it begin with

__?"), or about the book request itself (e.g., "Did you ask for a book title?"). Still other clarifications might elicit a change in request type (e.g., "What is the author's name?") or allow the wizard to select from elements of the search results (e.g., "Is the book title __?"). In Figure 4, the wizard chose to clarify that the caller had indeed said DIABETES (lines 8 – 9), and entirely ignored line 7.

The three database-query basic actions (by title, author, or catalog number) allow the wizard to do voice search. If any such action is selected, an additional small display (not shown) provides only the ASR output from the caller's responses for the current book request. To formulate the query, the wizard selects words from the small display with a mouse. The customized query then performs partial matching on the submitted ASR string against the book database. Wizards knew that database returns appeared in order of decreasing similarity, but the computed R/O scores did not appear on the GUI. A search by title or catalog number returned the top-scoring five books (title, author, and catalog number). A search by author returned up to 3 candidates each for up to 5 matching authors, a maximum of 15 search results. After line 9 in Figure 4, the wizard did a title search on PAGAN TURNER DIABETES. The five returns from the query, in descending R/O score order, were

Conquering Diabetes: A Cutting Edge Comprehensive Program for Prevention and Treatment? by Anne L Peters
Coping With Diabetes by Pat Kelly
Controle Su Diabetes Guia Para El Ciudado De Su Salud: Take Charge Of Your Diabetes A Guide For Care'? by US Dept of Health and Human Services
I'm Tougher than Diabetes by Alden R Carter
The Other Diabetes Living and Eating Well with Type 2 Diabetes by Elizabeth Hiser

From among these returns, the wizard incorrectly guessed the first one, and then asked for the author (lines 10 – 12). One confident word (SERVICES) in the ASR output was now enough to identify the book (lines 13-15). Finally, in line 16, the wizard selected a basic action.

Auxiliary actions were also provided. They allowed the wizard to remove a book from the order, to end the call, to ask if the caller was still there, to ask if the caller wanted to end the call, and to inform the caller that the book just requested had already been ordered during the current call.

Even with fully accurate ASR, to provide a careful match a wizard may use clarifications to choose judiciously among search results, as in Figure 5. In the first dialogue fragment (lines labeled A), the wizard did an author search on NANCY WHITELAW, but both the top two returns (out of five) were by Nancy Whitelaw. The wizard then asked for the catalogue number, which matched one of them, *They Wrote Their Own Headlines: American Women Journalists*. The second dialogue fragment (lines labeled B), shows that even when a clarification introduces errors, it can be of help. The wizard did an author search on JOHN EGERTON, but again the top two returns had that author. Rather than disambiguate between a 15-word and a 9-word title, the wizard again requested the RC number. Although the ASR output was on two lines and ROHAN is not a digit, *Speak Now Against The Day: The Generation Before The*

Civil Rights Movement In The South has catalogue number 40546, so the four good digits sufficed to identify the book.

A1 W: Next book?
A2 C: NANCY WHITELAW
A3 W: Can I have the RC number?
A4 C: THREE EIGHT NINE EIGHT ZERO
A5 W: Is the title *They Wrote Their Own Headlines: American Women Journalists*?
A6 C: YES
A7 W: Would you like *They Wrote Their Own Headlines: American Women Journalists* by Nancy Whitelaw?
A8 C: YES

B2 W: Next book?
B2 C: JOHN EGERTON
B3 W: Can I have the RC number?
B4 C: FOUR
B5 C: .ROHAN. FIVE FOUR SIX
B6 W: Is the title *Speak Now Against the Day: The Generation before the Civil Rights Movement in the South*?
B7 C: YES
B8 W: Would you like *Speak Now Against the Day: The Generation before the Civil Rights Movement in the South* by John Egerton?
B9 C: YES
B10 W: Next book?

Fig. 5. Author and catalog number requests in two dialogue fragments. Both lead the wizard to the correct book, even though the second includes an ASR error.

During the experiment, the trainer remained in the room with the wizard, and could communicate with the caller via chat to coordinate breaks between calls and to restart the system if necessary. This facilitated the complex wizard-caller pair scheduling and dealt with any unforeseen difficulties. On the rare occasion of a system crash, the call was not preserved.

5.5 Data Collection

The knowledge acquired from the work reported here is ultimately intended to support the construction of an improved dialogue manager. The decisions any dialogue manager makes must be based on data available to the SDS at run-time. Therefore we collected 163 run-time features that describe the system, the caller's speech and the wizard's behavior, and characterize the dialogue:

• Features that describe each call, each caller utterance, and each *adjacency pair* (portion of a dialogue that began with a system prompt and ended just before the next

system prompt [26]). An adjacency pair contains one or more caller utterances and zero or more database searches.
• Features that describe the input and output of CheckItOut's ASR, natural language understanding, and confidence annotation modules.
• Features that describe the wizard's behavior, including her clickstream and queries.
• Features that describe dialogue cost, such as correct book identification and frequencies of non-understanding and misunderstanding.

Surveys were administered automatically to wizards on their first, 60th, and 120th calls. The survey allowed the wizards to report on their ease with and progress on the task, and elicited information about their strategies. Callers also completed surveys after their 15th, 30th, 60th, and 90th calls.

Table 2. Selected statistics from the full dialogue experiment. (Despite instructions to correct the system with "That's not what I said," on two calls five books were ordered.)

Dialogues	**μ**	**range**	**σ**
Ordered books	2.45	0 – 5	1.44
Correctly identified books	2.26	0 – 5	1.45
Utterances per call	22.36	4 – 40	5.06
Words per utterance	2.99	1 – 10	2.27
Queries per adjacency pair	1.09	1 – 6	0.33
Questions from wizard to caller per book request	3.41	0 – 9	2.49
Title length in words	5.96	1 – 34	4.38
Adjacency pairs with at least one database query	32%		
Fully successful calls	28%		
Failed calls (no books ordered)	17%		

Voice searches	By title	By author	By catalog number
Number	43%	31%	26%
Return includes correct book	28%	33%	58%

6 Results

6.1 The Dialogues

Ten callers (5 male, 5 female) each made 15 calls to each of 6 wizards (3 male, 3 female), for a targeted total of 900 calls. (We actually collected 913 dialogues due to the exigencies of data collection.) Each dialogue addressed the task posed in the scenario: to assume an identity and then order four books. The dialogues covered 3,394 book requests in all, and 20,415 caller utterances. There were 17,288 adjacency pairs, 32% of which contained at least one database query. A sample of the ASR indicated a WER of about 50%. Table 2 further summarizes the corpus.

Most important was that the wizards, despite their ablation, understood what the callers wanted. For the most part, wizards identified the books in the scenario: 92% of all ordered books (2.26 of the 2.45 per call) had actually been requested by the callers. Because they operated under the 6-minute time limit, wizards terminated 63% of all calls. Of the non-terminated calls, 76% were *fully successful* (all 4 books correctly identified and ordered) despite a WER of about 50%. This result confirms that wizards were able to identify the signal within the noise through context and appropriate interaction strategies.

Table 3. Comparative wizard performance. WA and WB were the two most proficient wizards; WE and WD were the least proficient.

Question type	***All wizards***	***WA***	***WB***	***WD***	***WE***
Signal non-understanding	37%	34%	42%	40%	33%
On ASR string	8%	2%	0%	12%	15%
On query results	36%	37%	40%	26%	32%
On requests	19%	27%	18%	22%	20%
Total	11562	2321	1540	2013	1868
Actions per call					
Question before any search	0.35	0.29	0.21	0.67	0.42
Explicit confirmations	6.07	6.76	4.18	5.32	6.59
Implicit confirmations	0.40	0.62	0.68	0.20	0.86
Move-on strategy	0.67	0.39	1.19	0.65	0.43
Actions per book request					
Database searches	1.77	2.10	1.72	1.70	1.70
Questions	3.41	4.09	2.28	3.68	3.90
Confirmations	**1.74**	2.05	1.10	1.56	2.36
Call statistics					
Fully successful calls	28%	33%	32%	24%	16%
Failed calls (no books ordered)	17%	7%	11%	16%	24%
Correct titles	2.26	2.69	2.54	2.05	1.09

Each dialogue represented considerable interaction with the caller (22.36 caller utterances). Ultimately, an autonomous agent that relies on strategies learned from this corpus should achieve the same success, yet at far lower cost (e.g., number of turns). Note too that the book titles in the scenarios were often considerably longer than what a traditional SDS elicits from its callers — the titles averaged about 6 words but ranged as high as 34. As one would expect, returns from catalog number queries to the database more often (58%) contained the requested book than queries by author (33%,) or title (28%). In the remainder of this paper, any cited difference is significant on a paired *t*-test at the 95% confidence level.

6.2 What Wizards Did

Among all searches, wizards queried 43% of the time by title, 31% by author and 26% by catalog number. When uncertain about the search results, wizards sometimes attempted more than one query, on different ASR substrings or with different search types. They averaged about one query per adjacency pair, but often searched on multiple ASR substrings, more for title searches than for searches by author or catalog number. When the correct book appeared among the search results, it was typically high on the list returned by the query: first 85% of the time, second 8% of the time, third 3%, and later 4%.

When uncertain about the ASR or query results on either patron identity or book requests, wizards asked questions. The nature of these 11,562 questions is summarized in Table 3. Only 1% of all questions came before the wizard had made any database query at all. Wizards could ask for explicit confirmation of a full concept (e.g., ask the caller to confirm the title with a yes/no answer, as in line 14 of Figure 4) or of part of a concept (e.g., ask the caller to confirm a single word with a yes/no answer, as in line 8 of Figure 4), or confirm implicitly (e.g., have the TTS speak the title and then ask for the next book).

6.3 Exceptional Wizardry

Among our six wizards, WA and WB were considerably more successful than the others, as indicated by Table 3. WA and WB identified the most correct books per call, and had the fewest failed calls among all the wizards. Despite these similarities, WA and WB displayed very different approaches to their task.

WA, our *service-driven wizard,* worked hard to understand the caller. Per book request, WA did more searches than any other wizard, and asked more questions than any other wizard. One of the actions available on the wizard GUI under non-understanding was *move on*, which told the caller that the system was having difficulty understanding the current book request, and that the caller should continue on to a different book request and return to this one later. Among all wizards, WA used move on least often. WA often asked for confirmation, with the second-highest confirmation rate per book request. WA wanted to be sure; 92% of WA's confirmations were explicit ones.

In contrast, we theorize that WB, our *data-driven wizard,* had a pragmatic strategy: obtain high-quality ASR output to use for queries, search from it once or twice, and then either identify a book quickly or move on. Our data support this theory. WB indicated non-understanding more often, presumably to get better ASR output. Unlike the poorly-performing WD, however, WB never once asked a question about ASR output. Instead, WB used it to search. (Four of the 6 wizards, including WB, searched 1.70 – 1.73 times per request.) Moreover, WB asked questions far less often than any other wizard. (Other than WB's 2.28, question frequency was 3.53 – 4.09 per book request.) More confident than the other wizards, WB confirmed the least often of any, and 14% of WB's confirmations were implicit. Finally, WB used the move-on strategy more than any other wizard, nearly twice as often as the next most frequent user.

6.4 Caller Impact

The caller population was deliberately varied to provide the wizards with a range of recognition difficulties. Table 4 offers some comparisons. The best caller, C1, had the most correctly identified books per call on average. In contrast, the two worst callers, C0 and C2, averaged about one correct book per call, and the wizards made more queries to try to help them. Nonetheless, C0 and C2 had hardly any fully successful calls, and more than 40% of their calls failed to order any books at all.

Our best caller, C1, was more readily understood. Speech from C1 had the best recognition across all request types. Whether the wizards searched for a title, an author, or a catalog number, C1 had the highest percentage of database returns that included the correct book. Indeed the book C1 requested was often returned by the first query. C1 required the fewest database queries per adjacency pair on average. C1's well-recognized speech also produced the shortest calls, both in number of utterances and in elapsed time.

In contrast, speech from C0 had the worst recognition among all callers, and had the most utterances per call. Caller performance was not correlated with utterance length, however. Among the 10 callers, C1 had the third fewest words per utterance, C2 had the third highest, and C0 the fifth. Speech from C2 had the worst recognition for titles and authors.

Table 4: Comparative caller performance. C1 was the most successful caller; C0 and C2 were the least successful.

	All callers	***C1***	***C0***	***C2***
Correct books per call	2.26	3.26	0.96	1.03
Fully successful calls	28%	63%	3%	5%
Failed calls (no books ordered	17%	6%	41%	43%
Wizard searched based on caller utterances by				
Title	43%	32%	44%	47%
Author	33%	27%	37%	35%
Catalog number	26%	41%	19%	18%
Wizard found the caller's book on searches by				
Title	28%	42%	12%	11%
Author	33%	55%	20%	18%
Catalog number	58%	77%	35%	44%
Queries per book request	1.77	1.45	2.05	2.01
Utterances per call	22.36	19.29	23.97	21.99
Words per utterance	2.99	2.82	2.98	3.11

C1 is male; C0 and C2 are female; all three are native speakers of English. (C0 and C1 have an Eastern seaboard regional accent; C2 has a very slight Indian English accent.) Demographic data collected prior to the experiment indicated that C1 is age 18–25, while C0 and C2 are age 25–35. All three have a relatively fluent speech quality, although C0's speech rate is slow. Among the other callers, one was a native

speaker of Korean, another of Spanish, and two more described themselves as bilingual.

Caller success was based on more than ASR quality, however. Recall that the caller was permitted to select how she wanted to request the fourth book (by title, author, or catalog number). C1, our best caller, not only had the highest percentage of correctly identified books across request type, but also preferred the most readily recognized query type — speech from C1 evoked the highest percentage of queries by catalog number, and the lowest percentage for title and author. In contrast, speech from the poorly performing C0 evoked the most queries by author of any caller. Recognition distribution was not uniform across callers, For example, C3's title and author searches were equally successful (30%), while C4's title searches returned more correct titles (38%) than did her author searches (30%).

Differences among callers also emerged in the surveys. C3 reported that the system had difficulty recognizing catalog numbers, and was better with titles and authors. C9 reported that the system recognized author names poorly and often mispronounced them.

7 Discussion

The next generation of autonomous agents should be able to conduct a dialogue with their callers to achieve a common goal. Those callers are unlikely to be in soundproof facilities, and so the agent will have to contend with noisy acoustic data. Even if careful engineering eventually achieves perfect ASR, that will not prevent the agent's confusion as its human partner mumbles or coughs her way through their conversation. The key, we believe, lies in a mixed initiative system with voice search, which permits the agent to apply knowledge and clarification dialogues to understand its caller.

7.1 Generality and Applicability

Because the fundamental features of spoken dialogue systems and the tolerance of their callers are fairly domain-independent, much of the work reported here is applicable to other domains where a rich knowledge base is readily available to provide the wizard with context. Even many of the clarifications and basic actions would readily transfer with a bit of rewording. A particularly helpful feature of this domain was an ability to describe the desired object (a book) in several ways: by title, author, or catalog number. There was no assumption that the description was unique, but ability to shift to another descriptive mode was definitely helpful.

A different domain, however, would require a thoughtfully-designed wizard GUI, and the scale of the endeavor would have to justify the design and collection efforts. As for interface design, the preliminary work in Section 4 helped identify which actions should be available on the GUI for the full dialogue experiment. Considerably less effort went into the GUI's layout; our focus was on problem solving, not speed or ergonomics. In our view, what is required for good wizard GUI design is expertise —

expertise not in the domain itself, but in wizardry within it. Several wizards commented in the survey that they would have liked two additional basic actions on their GUI: "thank you" and "sorry." We expect to include both in future experiments.

Asynchronous SDS architectures (e.g., [6, 27, 28]) are not hampered by the traditional pipeline. An asynchronous architecture can bring to bear features like those collected here to disambiguate human communication. An asynchronous SDS can also profit from the work reported here, which is fundamentally about modes of problem solving. We have under construction an asynchronous SDS architecture, *FORRSooth,* that interleaves spoken language understanding and dialogue management [29, 30]. Models derived from the full dialogue experiment are expected to identify additional features and provide guidance on their use in the new system. Indeed, simulation of some wizard behaviors may require asynchronicity. Often, while a wizard decides what action to take next, she may also update her beliefs about the book request, decide whether an ASR output line is worthy of attention at all, decide whether a request is for a title or an author or a catalog number, and estimate the utility of her choices. In a pipeline architecture, these interrelations for the most part must be ignored.

7.2 How Wizards Solved Problems

Regardless of the SDS architecture, every agent designer need not mount a full-scale experiment like the one reported here. We have detected two different but equally successful approaches for an autonomous agent that understands during dialogue. WA's service-oriented approach persists in its attempt to understand, and asks many questions. It uses voice search extensively, and regularly seeks confirmation (particularly explicit confirmation) from the caller. WB's data-driven approach is more focused on the goal, and also less chatty and determined. It has more confidence in its own decisions, and seeks less guidance and reinforcement from the caller. We suspect that its matching mechanism is more elaborate and that it would more readily accelerate problem solving, and look forward to studying it further.

The reader is well-advised to compare WA and WB in terms of throughput. WB is faster: 48% of WB's tasks ran over the 6-minute limit, while 61% of WA's did. WA spent the extra time asking questions and confirming results, while WB preferred to abandon problematic book requests quickly. (For this reason, the dialogues in Figures 4 and 5 were chosen from calls to WA. Transcripts of calls to WB are far less evocative.) WB processed 20% more book requests than WA did, and yet achieved the same accuracy. While WA's questions focused more on the book request, WB wanted search-worthy ASR output.

The two least successful wizards, WD and WE, had the fewest fully successful calls, and the fewest correct titles per call. (WE also had the most failed calls.) WD and WE did try hard; among all the wizards, they had by far the most calls that exceeded the 6-minute limit (70% and 79%, respectively). WD and WE, however, tried to understand the ASR with less reliance on voice search: 12% and 15% of their questions, respectively, concerned the ASR, compared to 0% – 6% for the other wizards. They also recorded the most questions per request before any database

search at all. Wizards who focused on the ASR's version of the callers' words, rather than on their intent, were less successful.

One can readily imagine situations in which one or the other of these equally successful approaches would be preferable, as reflected by SDS caller studies. A service-oriented agent would offer more support to the caller, and so would be more appropriate for novice callers, the elderly, the young, and in urgent situations. A data-driven agent like WB might suffice in other situations, for example, for expert callers. In addition, callers might idiosyncratically prefer one of the two wizard approaches to the other.

Users talk less to WB (who had the fewest user utterances per call of all wizards) and WB talks less to them (and also had the fewest questions per call of all wizards). WB's confidence, however, sometimes confused the callers. Two of them indicated by survey that when "the system" had asked for the next book without telling them that it had just identified one, they asked for a summary at the end of the order and were surprised to find that "it" had gotten it right. An implemented version of WB should probably confirm more often.

There are presumably many reasons for the differences between our two successful wizards. WA majored in linguistics as an undergraduate and is now a Masters student in computer science, while WB is an undergraduate in computer science. WA is female and WB is male. Although their behaviors are consistent with gender-based styles of verbal communication once noted in the sociolinguistic literature, that is no longer a widely-accepted result [31, 32]. Rather, it appears to us that WA and WB brought different skill sets and attitudes to the task, and used the GUI to exploit them.

We do not claim that there are only two ways to succeed at this experiment as a wizard, merely that we observed only two in our data. We sought to identify, and now intend to implement, simple, rapid mechanisms that would improve CheckItOut's accuracy in ways familiar and acceptable to its callers. We are training models of both WA and WB at this writing. An open question is which of them would be most appropriate. That choice may be application-dependent or even caller-dependent. We anticipate further experiments with both models.

7.3 System Performance

Some comparisons between the baseline and the full dialogue experiment are instructive. Our wizards had extensive information on the GUI to process, and, as expected, they did not respond in "real time" — callers, as forewarned, waited unnaturally long for a response. Wizards were far slower (required about twice as much time per call) than CheckItOut. Given the wizards' instructions and ASR output where about half the words were incorrect, however, the wizards' persistence and the callers' tolerance despite lengthy pauses were exceptional. Indeed, one caller who had also participated in the baseline experiment volunteered that he was proud to participate in an experiment where the software had shown such marked improvement!

Wizards had been asked to facilitate orders as best they could and, compared to the baseline, they understood more often. Wizards identified more books correctly than CheckItOut: 2.26 books per call compared to 2.15 for the baseline. Our two best

wizards did even better, with 2.69 and 2.54 books per call. Wizards misunderstood less often than CheckItOut: callers signaled misunderstanding ("That's not what I said") five times more often with the baseline than with the wizards (0.54 per call compared to 0.11 with the baseline). Moreover, wizards ordered the wrong books about four times less often: 0.18 per call compared to 0.79 with the baseline.

The next step is to build new dialogue managers for CheckItOut that model our most successful wizards. To do so, we will further mine the corpus developed here. Other work in this area has considered a small set of features (e.g., 10 in [33] and 17 in [16]). We successfully learned models from the preliminary book title experiment [22]. In that work we began with 60 features, and learned, for example, decision trees that successfully predicted when the best wizard would select a title (F = .91) or ask a question (F = .68). Learning from full dialogues is considerably more complex than learning from book titles alone, however. For the experiment recounted here, we extracted 163 features. We expect that different feature combinations best predict different wizard actions, and that feature selection informed by knowledge about SDS components will support learning the best models. To this end, we have developed a new, general method for feature selection prior to learning a wizard model and an SDS-specific modification to it [34]. Once learned, models of WA and WB and their particularly relevant features will provide decision rationales in a repertoire of competing strategies for FORRSooth (described in Section 7.1).

8 Conclusion

A Wizard of Oz study provides data that allows computer scientists to model an agent's conversational skill on people. The corpus developed here will be released to the research community in 2012. It is distinguished from other wizard studies (described in Section 3.3) by its size, its richness, and its real (rather than simulated) ASR. It is also noteworthy for its 163 features that describe the experience of the system (ASR, NLU, confidence annotator, backend), the experience of the wizard, and the dialogue history.

To support the next-generation of autonomous agents in human-machine dialogue, we have mined this corpus for insight into the ways that people manage to understand one another during dialogue. Contextual reference (here, as voice search) is clearly more useful than extensive attention to the words detected by the ASR. Repeated non-understandings make CheckItOut terminate a call, and repeated misunderstandings force the caller to repeat "That's not what I said." In contrast, our wizards asked questions, and thereby understood what the caller did and did not want more often. Of the two successful approaches for an agent identified here, one is primarily service oriented, and the other is more data driven. Together they serve as guidelines for how an agent should, and should not, converse with people.

Acknowledgments. This research was supported in part by the National Science Foundation under awards IIS-084966, IIS-0745369, and IIS-0744904.

References

1. Levin, E. Passonneau, R.: A Woz Variant with Contrastive Conditions. In: Interspeech Satelite Workshop, Dialogue on Dialogues: Multidisciplinary Evaluation of Speech-based Interactive Systems (2006)
2. Passonneau, R.J., Epstein, S.L., Ligorio, T., Gordon, J. Bhutada, P.: Learning About Voice Search for Spoken Dialogue Systems. In: 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010), pp. 840-848 (2010)
3. Bohus, D. Rudnicky, A.: The Ravenclaw Dialogue Management Framework: Architecture and Systems. Computers in Speech and Language. 23, 332-361 (2009)
4. Raux, A., Langner, B., Black, A. Eskenazi, M.: Let's Go Public! Taking a Spoken Dialog System to the Real World. In: Interspeech 2005 (Eurospeech) (2005)
5. Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P. Zue, V.: Galaxy Ii: A Reference Architecture for Conversational System Development. In: 5th International Conference on Spoken Language Systems (ICSLP-98) (1998)
6. Raux, A. Eskenazi, M.: A Multi-Layer Architecture for Semi-Synchronous Event-Driven Dialogue Management. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2007) (2007)
7. Raux, A. Eskenazi, M.: Optimizing Endpointing Thresholds Using Dialogue Features in a Spoken Dialogue System,,. In: SIGdial 2008 (2008)
8. Huggins-Daines, D., Kumar, M., Chan, A., Black, A.W., Ravishankar, M. Rudnicky, A.: Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 185-189 (2008)
9. Ward, W. Issar, S.: Recent Improvements in the Cmu Spoken Language Understanding System. In: ARPA Human Language Technology Workshop, pp. 213-216 (1994)
10. Bohus, D. Rudnicky, A.: Integrating Multiple Knowledge Sources for Utterance-Level Confidence Anno-Tation in the Cmu Communicator Spoken Dialogue System. . Technical report, Carnegie Mellon University (2002)
11. Swift™: Small Footprint Text-to-Speech Synthesizer, Http://Www.Cepstral.Com/,
12. Bohus, D.: Error Awareness and Recovery in Task-Oriented Spoken Dialogue Systems (2004)
13. Stoyanchev, S. Stent, A.: Predicting Concept Types in User Corrections in Dialogue. In: EACL Workshop SRSL, pp. 42-49 (2009)
14. Litman, D., Hirschberg, J. Swerts, M.: Characterizing and Predicting Corrections in Spoken Dialogue Systems. Computational Linguistics. 32, 417-438 (2006)
15. Dix, A., Finlay, J., Abowd, G.D. Beale, R.: Human-Computer Interaction. Prentice Hall, (2003)
16. Rieser, V. Lemon, O.: Using Machine Learning to Explore Human Multimodal Clarification Strategies. In: COLING/ACL-06, pp. 659-666 (2006)
17. Skantze, G.: Exploring Human Error Recovery Strategies: Implications for Spoken Dialogue Systems Speech Communication, Special Issue on Speech Annotation and Corpus Tools. 45, 207-359 (2005)
18. Rieser, V., Kruijff-Korbayová, I. Lemon, O.: A Corpus Collection and Annotation Framework for Learning Multimodal Clarification Strategies. In: Sixth SIGdial Workshop on Discourse and Dialogue, pp. 97-106 (2005)
19. Sherwani, J., Yu, D., Paek, T., Czerwinski, M. Acero, A.: Voicepedia: Towards Speech-Based Access to Unstructured Information. In: Interspeech 2007 (2007)
20. Gordon, J.B. Passonneau, R.J.: An Evaluation Framework for Natural Language Understanding in Spoken Dialogue Systems. In: Seventh International Conference on

International Language Resources and Evaluation (LREC '10). European Language Resources Association (ELRA) (2010)
21. Passonneau, R., Epstein, S.L. Gordon, J.B.: Help Me Understand You: Addressing the Speech Recognition Bottleneck. In: AAAI Spring Symposium on Agents that Learn from Human Teachers. AAAI (2009)
22. Ligorio, T., Epstein, S.L., Passonneau, R.J. Gordon, J.B.: What You Did and Didn't Mean: Noise, Context, and Human Skill. In: Cognitive Science - 2010 (2010)
23. Passonneau, R.J., Epstein, S.L., Gordon, J.B. Ligorio, T.: Seeing What You Said: How Wizards Use Voice Search Results. In: IJCAI-09 Workshop on Knowledge and Reasoning in Practical Dialogue Systems. AAAI Press (2009)
24. Ratcliff, J.W. Metzener, D.: Pattern Matching: The Gestalt Approach, Dr. Dobb's Journal. (1988)
25. Bangalore, S., Boulllier, P., Nasr, A., Rambow, O. Sagot, B.: Mica: A Probabilistic Dependency Parser Based on Tree Insertion Grammars. In: NAACL HLT 2009 Companion Volume: Short Papers, pp. 185-188 (2009)
26. Sacks, H., Schegloff, E.A. Jefferson, G.: A Simplest Systematics for the Organization of Turn-Taking for Conversation. Language. 50, 696-735 (1974)
27. Allen, J., Ferguson, G. Stent, A.: An Architecture for More Realistic Conversational Systems. In: 6th International Conference on Intelligent User Interfaces, pp. 1-8 (2001)
28. Skantze, G. Gustafson, J.: Attention and Interaction Control in a Human-Human- Computer Dialogue Setting. In: Tenth Annual Meeting of the Special Interest Group in Dialogue and Discourse (SIGDIAL 10), pp. 310-313 (2009)
29. Gordon, J.B., Passonneau, R.J. Epstein, S.L.: Helping Agents Help Their Users Despite Imperfect Speech Recognition. In: AAAI Symposium Help Me Help You: Bridging the Gaps in Human-Agent Collaboration (2011)
30. Gordon, J., Epstein, S.L. Passonneau, R.J.: Learning to Balance Grounding Rationales for Dialogue Systems. in: 12th SIGDial on Dialogue and Discourse. (2011)
31. Cameron, D.: The Myth of Mars and Venus: Do Men and Women Really Speak Different Languages? Oxford, (2007)
32. Cameron, D.: Sex/Gender, Language and the New Biologism. Applied Linguistics. 31, 173-92 (2010)
33. Skantze, G. Edlund, J.: Early Error Detection on Word Level. in: ISCA Tutorial and Research Workshop on Robustness Issues in Conversational Interaction. (2004)
34. Ligorio, T.: Feature Selection for Error Detection and Recovery in Spoken Dialogue Systems. Ph.D. thesis, Computer Science, The Graduate Center of The City University of New York, New York (2011)