Toward Spoken Dialogue as Mutual Agreement

Susan L. Epstein[1,2], Joshua Gordon[4], Rebecca Passonneau[3], and Tiziana Ligorio[2]

[1]Hunter College and [2]The Graduate Center of The City University of New York, New York, NY USA
[3]Center for Computational Learning Systems and [4]Department of Computer Science, Columbia University New York NY USA
susan.epstein@hunter.cuny.edu, becky@cs.columbia.edu, joshua@cs.columbia.edu, tligorio@gc.cuny.edu

Abstract

This paper re-envisions human-machine dialogue as a set of mutual agreements between a person and a computer. The intention is to provide the person with a habitable experience that accomplishes her goals, and to provide the computer with sufficient flexibility and intuition to support them. The application domain is particularly challenging: for its vocabulary size, for the number and variety of its speakers, and for the complexity and number of the possible instantiations of the objects under discussion. The brittle performance of a traditional spoken dialogue system in such a domain motivates the design of a new, more robust social system, one where dialogue is necessarily represented on a variety of different levels.

Introduction

A spoken dialogue system (*SDS*) has a social role: it supposedly allows people to communicate with a computer in ordinary language. A *robust* SDS should support coherent and habitable dialogue, even when it confronts situations for which it has no explicit pre-specified behavior. To ensure robust task completion, however, SDS designers typically produce systems that make a sequence of rigid demands on the user, and thereby lose any semblance of natural dialogue. The thesis of our work is that a dialogue should evolve as a set of agreements that arise from joint goals and the collaboration of communicative interaction (Clark and Schaefer, 1989). The role of metacognition here is to use both self-knowledge and learning to represent dialogue and to enhance the SDS. As a result, dialogue should become both more habitable for the person and more successful for the computer. This paper discusses the challenges for an SDS in an ambitious domain, and describes a new, metacognitively-oriented system under development to address the issues that arise in human-machine dialogue.

Our domain of investigation is the Heiskell Library for the Blind and Visually Impaired, a branch of The New York Public Library and part of The Library of Congress. Heiskell's patrons order their books by telephone, during conversation with a librarian. The volume of calls from its 5028 active patrons, however, promises to outstrip the service currently provided by its 5 librarians.

The next section of this paper describes the challenges inherent in spoken dialogue systems. Subsequent sections describe a traditional SDS architecture, demonstrate the brittle behavior of an SDS built within it, and re-envision a new SDS within the structure of a cognitively-plausible architecture. The paper then posits a paradigm that endows human-machine dialogue with metacognition, explains how metacognition is implemented in this re-envisioned system, and reports on the current state of its development.

Challenges in SDS Implementation

The social and collaborative nature of dialogue challenges an SDS in many ways. The spontaneity of dialogue gives rise to *disfluencies*, where a person repeats or interrupts herself, produces filled pauses or false starts and self-repairs. Disfluencies play a fundamental role in dialogue, as signals for turn-taking (Gravano, 2009; Sacks, Schegloff and Jefferson, 1974) and for *grounding* to establish shared beliefs about the current state of mutual understanding (Clark and Schaefer, 1989). Most SDSs handle the content of the user's utterances, but do not fully integrate the way they address utterance meaning, disfluencies, turn-taking and the collaborative nature of grounding.

During dialogue, people simultaneously manage turn-taking and process speech. The complexity of speech recognition for multiple speakers, however, requires the SDS to have an a priori dialogue strategy that determines how much freedom it offers the user. An SDS that maintains *system initiative* completely controls the path of the dialogue, and dictates what the person may or may not say during her turn. ("SAY 1 FOR ORDERS, SAY 2 FOR CUSTOMER SERVICE, OR…"). In contrast, habitable dialogue requires *mixed initiative*, where the user and the system share control of the path the dialogue takes. Of course, mixed initiative runs the risk that the system will find itself in a state unanticipated by its designer, and no longer respond effectively and collaboratively. Because fallback responses (e.g., asking the user to repeat or start over) are brittle, current mixed-initiative systems pre-specify how much initiative a user may take, and restrict that initiative to specific kinds of communicative acts.

An SDS receives a continuous stream of acoustic data. Automated Speech Recognition (*ASR*) translates it into discrete linguistic units (e.g., words and phonemes) represented as text strings. Such continuous speech recognition over a large vocabulary for arbitrary speakers presents a major challenge. The Heiskell Library task includes 47,665 distinct words from titles and author names, with a target user population that varies in gender, regional accent, native language, and age. Moreover, telephone speech is subject to imperfect transmission quality and background noise. For example, the word error rate (*WER*) for Let's Go Public! (Raux et al., 2005) went from 17% under controlled conditions to 68% in the fielded version.

Speech engineering for a specific application can reduce WER, but dialogue requires more than perfect transcription; it requires both the speaker's meaning and her intent. Once it has recognized the other's intent, a dialogue participant must also respond appropriately. An SDS tries to confirm its understanding with the user through the kinds of grounding behaviors people use with one another. Repetition of the other's words, along with a request for agreement, is a traditional form of grounding, albeit annoying in an SDS. An SDS that reports, "I HEARD YOU SAY *THE GRAPES OF WRATH*. IS THAT CORRECT?" seeks *explicit confirmation* for its ASR output. Although explicit confirmation guarantees that the ASR transcribed the sound correctly, it soon annoys the user. *Implicit confirmation* (e.g., "STEINBECK IS A POPULAR AUTHOR"), or even no confirmation at all, makes conversation more habitable. Yet any grounding other than explicit confirmation runs the risk that the SDS will misunderstand the user, and thereby compromise its correctness.

Finally, a habitable SDS must understand turn-taking behaviors, including when the user wants to interrupt and seize the next turn, and when the user is willing to cede the current turn. An SDS that allows mixed initiative must still rely on simplistic approaches to turn-taking because it cannot distinguish between a signal that the user is still listening) and a genuine confirmation. This limits the range of grounding behaviors that can be implemented.

A Traditional SDS Architecture

Many contemporary SDSs have a pipeline-like architecture similar to that of *Olympus,* shown in Figure 1 (Bohus et al., 2007; Bohus and Rudnicky, 2003). The person at the left provides spoken input. As segments of acoustic data are completed, the audio manager (Raux and Eskenazi, 2007) forwards them to the ASR module, which transcribes the speech segment into a text string of words from its vocabulary. The text string is forwarded to the natural language understanding (*NLU*) module, which produces one or more semantic representations of it. The NLU identifies the objects of interest and their likely values. For example, the NLU might identify the string "SAMUEL COLERIDGE*"* either as the title of a biography or as an author, and the string "I'D LIKE *TO STOP NOW"* as either a request to terminate the dialogue or as a book request. Together, the ASR and the NLU *interpret* what has been said.

The NLU forwards the semantic representations it constructs to a confidence annotator, which scores them. Scoring is based on a variety of knowledge sources, including ASR confidence scores on the individual words, and how many words could not be included in the semantic interpretation. The highest-scoring interpretation is forwarded to the dialog manager, which determines what to do next. A strong match to data in a knowledge source supports and completes the semantic interpretation. In *CheckItOut*, the system we constructed for the Heiskell task within the Olympus framework, the *RavenClaw* dialogue manager may request information from its Domain Reasoner (*DR*) module. CheckItOut's DR queries a knowledge source backend with the semantic representation.

CheckItOut relies on *Phoenix*, a semantic parser for NLU that implements a set of context-free grammars (*CFGs*). Because it can omit words in the ASR from the final parse, Phoenix is robust to recognition errors. The rules for CheckItOut's book title CFG were automatically produced from full syntactic parses of Heiskell's book titles, with a broad coverage dependency parser (Bangalore et al., 2009). Rules modeled on syntactic parses provide linguistically-motivated constraints on word order, parts of speech, and constituent structure. They also make meaningful parses that are robust to misrecognized words more likely.

When the WER is high, the text string passed to the DR may match several choices in the backend nearly as well. For example, the ASR string "ROLL DWELL" elicited three returns: CROMWELL, ROBERT LOWELL, and ROAD TO WEALTH. Whether or not it uses the DR, the dialog manager eventually decides what to communicate to the user. That decision is forwarded to the natural language generator (*NLG*), which uses templates to produce text. For example, the dialogue manager might decide to confirm one of the titles returned from the database, and the NLG might use the template "Did you say *x*?" to produce "DID YOU

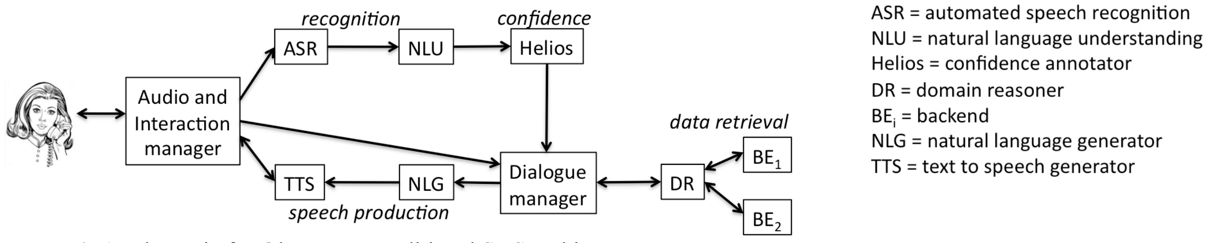
recognition
confidence
ASR
NLU
Helios
Audio and Interaction manager
data retrieval
BE$_1$
Dialogue manager
DR
TTS
NLG
BE$_2$
speech production
ASR = automated speech recognition
NLU = natural language understanding
Helios = confidence annotator
DR = domain reasoner
BE$_i$ = backend
NLG = natural language generator
TTS = text to speech generator

Figure 1: A schematic for Olympus, a traditional SDS architecture.

SAY 'ROAD TO WEALTH'?" The text-to-speech (*TTS*) module then transforms the text from the NLG into speech, the system's spoken participation in dialogue.

Brittle Behaviors from an SDS

The examples in this section are portions of dialogues from CheckItOut, one of 12 research dialogue systems built at half a dozen sites using the Olympus/RavenClaw framework. Each of them required the construction of a domain-specific dialogue task tree to guide its decisions, and a domain-specific DR if they accessed a backend database. (CheckItOut's databases detail Heiskell's books, anonymized patrons, and their transaction history). As an Olympus/RavenClaw application, CheckItOut also has access to domain-independent error handling.

Users may order as many as four books, and do so by book number, title, or author (for which CheckItOut then offers a choice of popular titles). The difficulties CheckItOut encounters motivated the metacognitive approach that is the focus of this paper. In the examples below, CheckItOut's generated speech appears in italics, with punctuation added for clarity.

Example 1: Too much separation between speech recognition and semantic processing

NEXT BOOK?

.ALLAN. STEELE

DID YOU SAY SHELBY STEELE?

NO

I'M SORRY. I MUST HAVE MISUNDERSTOOD YOU. NEXT BOOK?

.ALLAN. STEELE

DID YOU SAY SHELBY STEELE?

CheckItOut tries to recognize only the intended concept, based on the semantic interpretation of the current recognition hypothesis. The periods indicate low recognition confidence on ALLAN; it does not appear in the semantic interpretation. STEELE parsed as an author name, however, so the backend returned the most prolific Steele: Shelby. CheckItOut is unaware when a second recognition hypothesis is identical to the first. A metacognitive system with the appropriate data structures could respond differently when user repetition accompanies low confidence in recognition.

Example 2: Inability to backtrack to arbitrary points in the dialogue. If the system has misunderstood, users are told to respond with "That is not what I said."

NEXT BOOK?

THAT IS NOT WHAT

"THIS JUST IN: WHAT I COULDN'T TELL YOU ON TV" IS AVAILABLE.

THAT IS NOT LITERALLY

"I'M NOT REALLY HERE" IS AVAILABLE. NEXT BOOK?

THAT IS NOT LITERALLY

I'M SORRY. I THOUGHT YOU SAID, "I'M NOT REALLY HERE," BUT YOU ALREADY ORDERED IT. NEXT BOOK?

The user now has two books she did not want because the "wrong book" signal has been misunderstood three times as a title. Now she must either accept the error or start the dialogue over. Backtracking to redo a portion of the dialogue depends on both knowledge of the specific dialogue task structure and on domain-independent knowledge about repairing misunderstandings. RavenClaw provides domain-independent error handling strategies, and offers support for limited restructuring of its agenda. Its DM, however, only has limited information about recent utterances. At each step in the pipeline from ASR to NLU to DM, information is lost that may have been necessary to make good decisions. (For example, the speech rate of the utterance is unavailable to the DM.) Here, CheckItOut cannot tell whether it has misunderstood the speaker's words (here, recognizing the wrong title) or her intent (here, telling CheckItOut that it has made an error).

Example 3: Inability to perform incremental semantic interpretation. In the early part of the dialogue, the system identifies a user by her area code and telephone number. (CheckItOut does not use caller ID.)

CAN I HAVE THE PHONE NUMBER?

NINE NINE FIVE

SORRY, I DIDN'T UNDERSTAND YOU.

SEVEN FOUR THREE THREE

I'M TRYING TO IDENTIFY YOU. I HAVE YOUR AREA CODE. PLEASE SAY YOUR PHONE NUMBER.

NINE NINE FIVE SEVEN FOUR

SORRY, I DIDN'T UNDERSTAND YOU…

The ASR had captured different portions of the 8-digit telephone number. It could have composed the first two responses, or certainly composed all three, to produce a reasonable (and correct) guess: 995-7433. It cannot, however, move incrementally toward an interpretation of the first user utterance.

Example 4: Over-reliance on sequential processing before access to domain knowledge. Unlike most spoken dialogue systems, CheckItOut relies on voice search. This makes it possible to leverage domain knowledge for resolving noisy ASR (Passonneau et al., 2010a). In the Olympus/RavenClaw architecture, however, NLU precedes backend database queries, so CheckItOut applies a variant of voice search after parsing.

NEXT BOOK?

THUD A DEVIL WHERE SPADA

I DIDN'T UNDERSTAND YOU. PLEASE SAY THE NAME OF THE BOOK.

Here the NLU produced a single interpretation with two concepts: a book title beginning "THUD," and an author named SPADA. Neither concept produces a match under voice search. If the full text string had been submitted to voice search prior to NLU, however, "*The Devil Wears Prada*" would have been returned, correctly.

A social system must not only be correct but also consider the impression it makes upon the user. The PARADISE framework for dialogue evaluation models user satisfaction (measured, for example, by user questionnaires) (Walker, 1997). Its metrics address both task success and costs, such as number of turns, number of corrections from the user, and number of rejections by the user. By those standards, CheckItOut certainly warrants improvement. By its lack of reference to dialogue history and its inability to piece information together, CheckItOut appears inattentive to the conversation as a whole. Because of the pipeline, CheckItOut may overlook reasonable alternatives and be unable to retreat to others when its first choice fails. The resultant errors frustrate the user and make the system brittle.

Re-envisioning the SDS

This section envisions an SDS that is responsive to a broad range of WERs. The input to this system is knowledge from the backends, acoustic energy (speech) from the user, and confirmations of speech fragments from the system that went uninterrupted. System output is from the TTS.

Rather than focus on what it needs from the user to accomplish its task, this new system will support the social and collaborative nature of dialogue. Rather than box functions into separate modules as in Figure 1, its processes may execute in parallel and collaborate with or interrupt one another. Like a person, the resultant system will listen and interpret at once, anticipate, and process interruptions, all to achieve agreements with the user. Here, an *agreement* binds a value to a variable of interest (e.g., an area code), and dialogue is envisioned as exchanges that arrive at a set of mutual agreements.

Our proposed SDS has metaknowledge about dialogue. It knows that it is engaged in dialogue with another speaker, and that speakers take turns. It also knows the dialogue's *history* (record of what has transpired thus far), and has an *agenda* (a pre-specified set of agreements). Each agreement may be thought of as a subdialogue, and the agenda may be fully or partially ordered. For example, the library agenda has agreements for participation in the dialogue, user identification, some number of book requests, an order summary, and a farewell. The SDS maintains the agenda, and represents each agreement as one or more *targets*, items on which to agree. For example, the targets for the user identification agreement are area code, telephone number, name, and address. When all the targets in an agreement have been met, the SDS selects another agreement. When the entire agenda has been satisfied, the SDS terminates the dialogue.

Ideally, a target is satisfied by a single pair of *turns*, one for the SDS and one for the user. For example, the SDS requests an area code and the user provides it; or the user volunteers an area code, and the SDS knows what to do with it. Each turn has an *intent* (what it tries to convey) and an *expectation* (what it expects to hear). For example, when the system requests an area code, its intent is to ask a question and its expectation is that it will receive a valid one in its database. In turn, when the user says "212," her intent is to provide her area code, and her expectation is that the system will understand what she has said.

To demonstrate that a social agreement has been reached, the SDS must provide evidence to the user of its interpretations, and accept evidence from the user of hers. (Since it manages the agenda, the SDS always knows its own intent and expectation, but it must infer the user's intent.) After each user turn, the SDS compares its expectation from its own previous turn to the most recent ASR output. When that expectation has been met, the SDS grounds the target binding and then selects the next target in the agreement. When that expectation is not met, the SDS sets aside the agenda until the discrepancy is resolved.

Our new SDS, *FORRSooth,* provides all the functionality of a traditional SDS. In a spirit similar to Olympus, we provide modular interfaces for internal components, including speech recognizers and synthesizers. In this paradigm, however, interaction management, recognition, understanding, confidence, decision making, domain reasoning, text generation, and speech production are no longer sequential. Instead, they are interleaved with the assistance of a cognitively-plausible architecture.

FORR and FORRSooth

FORR (FOr the Right Reasons) is a domain-independent, architecture for learning and problem solving (Epstein, 1994). FORR is intended for a domain in which a sequence of decisions solves a problem. Robust and effective FORR-based systems include *Hoyle* the game learner (Epstein, 2001), *Ariadne* the simulated pathfinder (Epstein, 1998), and *ACE* the constraint solver (Epstein, Freuder and Wallace, 2005). Each of them is intended for a particular domain, such as game playing or pathfinding. FORRSooth is a FORR-based SDS, one intended for dialogue.

Knowledge

A FORR-based system relies on knowledge to support its decision making. In addition to traditional knowledge bases (e.g., the backend in CheckItOut), a FORR-based program uses descriptives. A *descriptive* is a shared data structure that is computed on demand, refreshed only when required, and referenced by one or more reasoning procedures. Some descriptives (e.g., time on task) are computed by FORR itself. Most descriptives, however, are domain-dependent. For the dialogue domain, these include the dialogue-specific metaknowledge described above: the dialogue history, the agenda, the agreements, their targets, and turn-taking. There are also descriptives for text strings from the ASR, parses, confidence levels, and backend returns. Others include user satisfaction, system accuracy, and computation times.

FORR enables FORRSooth to have a set of alternative actions under consideration. This permits FORRSooth to entertain multiple hypotheses about what the user said simultaneously. The agenda determines the kind of actions to be considered at any point in the dialogue. For example, if FORRSooth has just received confirmation of its current expectation, it can choose among a variety of grounding actions.

Decision making

Another reason that FORRSooth is FORR-based is that it is impossible to specify in advance the correct response to every user utterance. Instead, FORR combines the output from a set of domain-specific procedures called *Advisors* to decide how to respond. Each Advisor embodies a rationale for a particular kind of decision: matching, grounding, or error handling. Examples appear in Table 1. In Figure 1, the dialogue manager made these decisions alone, typically with a fixed set of rules or a function learned offline.

FORR organizes its Advisors into a 3-tier hierarchy. Tier-1 Advisors are reactive and guaranteed to be correct, such as *Perfect* and *Implicit-1* in Table 1. Tier-1 Advisors relevant to the decision type (matching, grounding, or error handling) are consulted in the order specified by the system designer.

Tier-2 Advisors are situation-based, that is, they respond to a pre-specified trigger. For example, in FORRSooth the trigger "expectation not met three consecutive times on this target" could alert some tier-2 Advisors that manage error handling. Once triggered, a tier-2 Advisor may specify a (possibly partially ordered) set of targets. Examples include *AlternativeID* and *Assemble* in Table 1.

Tier-3 Advisors are heuristics; they are consulted together and their opinions are combined to produce a decision. Output from a tier-3 Advisor is a set of *comments*, each of which pairs an action with a *strength* that indicates support for or opposition to that action. Note, for example, the variety of rationales in Table 1 that support particular backend returns from voice search. An Advisor may produce any number of comments, each on a different action.

When FORRSooth decides to speak, its agenda provides the current target to the hierarchy of Advisors, and indicates whether it is time to match or ground. The Advisors decide what to say. If any tier-1 Advisor can do so, no further Advisors are consulted and that action is taken. For example, *Implicit-1* might decide to say "WE HAVE THAT." If no tier-1 Advisor determines an action, control moves to tier 2. When a triggered tier-2 Advisor produces a set of targets (a *subdialogue*), it includes instructions on when to terminate the subdialogue. The system then revises the agenda to make the subdialogue its top priority. After any such revision, the hierarchy is consulted again from the top. The tier-1 Advisor *Enforcer* ensures appropriate subdialogue execution, suspension, and termination based upon instructions embedded in the subdialogue by its tier-2 creator. Finally, if neither tier 1 nor tier 2 makes a decision, control passes to tier 3. Tier-3 Advisors are likely to disagree on what to do. Conflicts among them are resolved by *voting*, which tallies a weighted combination of comment strengths for and against each action. The action with the highest score is chosen. Advisors' weights are learned.

FORR's Advisor hierarchy is a highly modular structure. It is easy to add Advisors as decision-making rationales are identified. (Thus far, the vast majority of FORRSooth's Advisors are dialogue-specific, but not application-specific, that is, they would serve for applications other than the Heiskell Library task.) Moreover, the rationales that underlie individual Advisors reflect behaviors we have observed when people succeed at a similar task, as described in the next section.

Human Skill Influences SDS Design

FORRSooth was inspired by behavior observed when people matched ASR output to book titles (Passonneau, Epstein and Gordon, 2009). Undergraduates were each given the ASR that resulted from 50 titles spoken by a single individual, along with a text file containing all 71,166 Heiskell titles. They were asked to match each ASR string to a title. There was no time limit, and they could search in any way they chose. Despite the fact that only 9% of the titles were rendered correctly by the ASR, the subjects' accuracy ranged from 67.7% to 71.7%.

Table 1: Some of FORRSooth's Advisors. Only those with an asterisk (*) are specific to the Heiskell Library task.

Tier	Advisor	Decision type	Rationale
1	*Perfect*	Match	ASR string had a perfect match from the backend, so return it.
1	*Implicit-1*	Ground	ASR string had a perfect match from the backend, so ground implicitly.
1	*Enforcer*	All	If a subdialogue exists, process it.
1	*NoRepeat*	Error handling	Same utterance twice in a row, so do not ask the user to repeat.
2	*Assemble*	Match	> 2 attempts on target, so guess combinations of those responses.
2	*AlternativeID*	Error handling	> 3 consecutive non-understandings, so ask the user for the author or number.
2	*NotWhatSaid*	Error handling	If "that's not what I said," reconsider recent variable bindings in reverse order.
3	*Popular*	Match	Select returns from the backend with the highest circulation frequency.
3	*FavoriteGenre*	Match	*Select books with the user's favorite genre.
3	*FavoriteAuthor*	Match	*Select books by the user's favorite author.
3	*SoundsLike*	Match	The return sounds like the ASR text string.
3	*SpelledLike*	Match	The return is spelled like the ASR text string.
3	*FirstWord*	Match	The return matches the first word in the ASR text string.
3	*JustMatch*	Match	The return matches the ASR text string.
3	*Parse*	Match	The return matches a parse.
3	*UnusualWord*	Match	The return contains an unusual word in the ASR text string.
3	*Explicit-3*	Ground	This was difficult to understand, so ground explicitly.
3	*Implicit-3*	Ground	This dialogue is unusually long, so ground implicitly.

In a second experiment, we sought to understand the mechanism underlying that skill. In this experiment, pairs of undergraduate computer science majors spoke Heiskell book titles to one another through a speech recognizer. One person played the role of user and the other was the subject. The experiment was designed to make the speech more like dialogue than the reading of a list. For further details, see (Passonneau et al., 2010b).

The subject sat at a graphical user interface and served as the dialogue manager in Figure 1. She could see the ASR string and could query the full Heiskell database with it. (To evaluate the quality of a match against the ASR, we adapted the Ratcliff/Obershelp similarity metric: the ratio *r* of the number of matching characters to the total length of both strings (Ratcliff and Metzener, 1988).) Up to 10 matches, in descending order by (concealed) match score were displayed in response to a query. Words in the returns that matched a query word appeared darker on the screen. The subject was then expected, in real time, to select the title that had been requested, ask a question that might help her choose, or give up on matching that request. Over several weeks, each of the seven subjects requested 100 titles from every other subject, 4200 title requests in all.

Had a subject simply selected the first (i.e., top-scoring) return, she would have been accurate 65% of the time. Our subjects' accuracy, however, ranged from 69.5% to 85.5%. To find rationales for our Advisors, we sought the factors that supported our subjects' decisions. We extracted training samples from the data, and learned decision trees that modeled individual subjects' actions well ($0.60 \leq F \leq 0.89$). Linear regression and logistic regression models had similar results. Key features in these trees will become Advisor rationales: the number and scores of the returns, the frequency with which the subject had been correct on the last three titles, the maximum number of contiguous exact word matches between a return and the ASR string (averaged across candidates), and the Helios confidence score. (Confidence scores, metrics on matches, and success on titles other than the last one did not appear on the GUI.) The tree for our top-scoring subject also used the length of the ASR string first to choose a decision strategy.

Metacognition for an SDS

Our version of the paradigm for metacognition established by Cox and Raja (Cox and Raja, 2007) appears in Figure 2. Only the speech from the user, the speech from the system, and the backend returns lie at the object level. Knowledge about that speech, represented as descriptives, supports both reasoning at the object level and metareasoning. The object layer contains FORRSooth's Advisors for grounding, matching, and error handling. Grounding strategies range from simple confirmations to subdialogues. Based on dialogue confidence and history, tier-1 Advisors make fast and obvious decisions, tier-2 Advisors propose clarification dialogues, and tier-3 Advisors support a specific action. In this way, easy choices are made quickly, and difficult ones take a little longer.

FORRSooth has a clear metacognitive orientation, that is, it reasons about which decision algorithms to use and about its own level of understanding. The metacognitive features of our re-envisioned SDS replace the "react to speech" paradigm of Figure 1 with "establish a set of mutual agreements." The comparison of expectation with response, and the determination to establish common beliefs provide metalevel control that gives error handling priority over the establishment of additional agreements.

By design, FORRSooth is mixed initiative. Its reactive interaction manager mediates between the continuous, realtime nature of dialogue and the discrete reasoning of the Advisors in the object layer. The interaction manager transmits utterances between the user and the system. It also updates the descriptive for spoken input when the user stops speaking.

The second experiment above provided clear evidence that a robust SDS requires awareness of both its performance and its knowledge. For performance, recall that our subjects consistently used their recent task success to make choices. An SDS should gauge and use its *self-confidence*, as measured by system accuracy and user feedback on the last *n* requests or dialogues. There are many plausible ways to integrate self–confidence into Advisors in every category. For example, it can be a factor for consideration in tier 3, or mandate more caution than would otherwise be exercised in tier 1. Metareasoning can also address confidence in individual values. For example, the way a decision is grounded should depend in part upon the confidence with which the match was made.

Another form of metacognition is knowing when you do not know. This explains the striking difference in the second experiment between our two most proficient subjects (85.5% and 81.3%) and the other five (69.5% to 73.46%). The two more proficient subjects knew when to ask a question. When the query returns were all poor matches, these two asked questions far more often than the others.

Learning is essential in FORRSooth. The system will learn weights for its many (likely contradictory) tier-3 Advisors. The weight-learning algorithm will reward Advisors that support good decisions and penalize those that make poor ones. Reinforcement size will reflect Advisors' relative success modeled on criteria from PARADISE. In FORR, a *benchmark Advisor* for each kind of action makes random comments. Benchmark Advisors do not participate in decision-making, but they do acquire learned weights. After sufficient experience, FORRSooth will not consult any Advisor whose weight remains consistently lower than that of its benchmark.

FORRSooth has many derived descriptives, but only

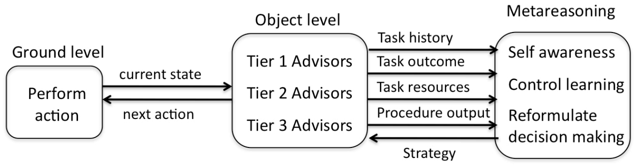

Figure 2: Metacognition and the FORR architecture.

three for ground-level information: speech from the user, uninterrupted speech from the system, and backend matches. Its other descriptives serve the reasoning level about what to say next, and support metareasoning about the Advisors and how they are organized. These descriptives include the agenda (whose default value is the set of agreements and their targets), the task history, and whose turn it is to speak, as well as confidence measures, Advisor weights, and various computations based on user input and the backend data (e.g., possible matches or parses). As weights are learned, a descriptive no longer referenced by any Advisor is no longer computed. Thus, the SDS gauges and exploits the usefulness of its own knowledge and rationales.

A FORR-based system is, by construction, boundedly rational. Advisors have a limited amount of time in which to construct their comments. FORR gauges their *utility* (accuracy per CPU second consumed). Weight learning can then consider utility as well as accuracy.

FORRSooth's dialogue proficiency will be gauged by task success and efficiency metrics similar to those of the PARADISE framework. The Advisors in Table 1 manage the difficulties raised earlier in this paper far better than CheckItOut did. *NoRepeat* addresses Example 1, *NotWhatSaid* handles Example 2, and *Assemble* deals with Example 3. Once appropriate weights are learned, Example 4 should be addressed by *JustMatch.*

Related and Future Work

Mixtures of heuristics have often been shown to enhance decision quality when they are weighted (Minton et al., 1995; Nareyek, 2003) or form a portfolio (Gagliolo and Schmidhuber, 2007; Gomes and Selman, 2001; Streeter, Golovin and Smith). FORR learns such a mixture, but it also learns which knowledge to compute to support it. Furthermore, it can reorganize its Advisors to speed its decisions (Epstein, Freuder and Wallace, 2005).

The dialogue manager of Figure 1 is a set of rules (in RavenClaw, represented as a tree) that anticipates paths a dialogue might take and relies on domain-independent error-handling protocols (Bohus and Raux, 2009). When the dialogue veers away from those predictions, the SDS becomes brittle. Rather than anticipate all possibilities, FORRSooth expects to learn appropriate behavior. Because they should impact one another, FORRSooth incorporates many functionalities of an SDS in addition to that of the traditional dialogue manager.

The traditional SDS's partition of hearing, reasoning, and speaking into separate components makes an integrated approach to reasoning and learning more difficult. As a result, machine learning has typically been restricted to the design phase. For example, some research has viewed dialog management as a Partially Observable Markov Decision Process, and learned a policy for it by reinforcement learning on a corpus (Levin, Pieraccini and Eckert, 2000; Williams and Young, 2007). In contrast, FORRSooth's metareasoning allows it to learn weights for its tier-3 Advisors online, so that it improves as it is used.

ALFRED, a task-oriented dialogue agent, addresses miscommunication from ambiguous references, including incompatible or contradictory user intentions and unknown words (Anderson, Josyula and Perlis, 2003). In contrast, FORRSooth manages non-understandings specific to spoken dialogue, particularly those stemming from recognizer noise or speech disfluency. Meta-reasoning in ALFRED is controlled by a formalism that augments inference rules with a constantly evolving measure of time. Knowledge about the environment, including perceptions of user utterances and the system's beliefs about those utterances, are represented in an associated knowledge base of first-order formulae. In contrast, FORR integrates multiple reasoning processes, and represents the passage of time as values for historical descriptives.

Matching Advisors consider parsing, voice search, and dialogue history. In a traditional SDS, the NLU maps the words to concepts. In FORRSooth, however, there are multiple descriptives (e.g., kind of utterance, possible parses, dialogue history) that a matching Advisor can reference to make a recommendation. (This is analogous to an NLU that employs multiple representations, such as (Gupta et al., 2006).) The tier-1 matching Advisor *Perfect* detects a perfectly matched title and returns it, without ever parsing. A tier-2 matching Advisor might trigger by some failure to understand, produce a subdialogue that combines multiple hypotheses from the dialogue history, and then ask "Did you mean *x*?" for each of them. A tier-3 matching Advisor could consider the number of possible parses or the voice search score (or some other rationale) to identify a match.

Domain-independent error-handling strategies in RavenClaw have been studied extensively (Bohus, 2007). That approach learns a confidence function for concepts from labeled training instances, updates its belief in only the current concept, and then either confirms the concept or repeats the error handling. In FORRSooth, however, we expect to record confidence on many descriptives' values. FORRSooth's error handling includes reactive Tier-1 Advisors (e.g., *NoRepeat*), tier-2 Advisors that propose clarification dialogues (e.g., *AlternativeID*), and tier-3 heuristics. Those heuristics may comment to prompt the user to repeat or rephrase her last utterance, or select an alternative way to request the information.

There is deliberately no commitment in FORRSooth to a fully-ordered agenda or to fully-ordered targets for an agreement. This provides considerable tolerance for mixed initiative that might simplify the system's task. (For example, while the system is assembling guesses, the user could repeat a title for which the match score is good.) Subdialogues are paused and resumed in a similar fashion.

FORRSooth is intended to be an SDS, not a book-ordering system; only its backend and a few of its error-handling Advisors are domain-specific. Building an SDS in FORR allows the system designer to entertain multiple heuristic rationales, and permits the system to learn from its experience what would be a good combination of them for the task at hand. The focus of current development is weight learning based on PARADISE metrics, novel ways for the system to guess at what a user means (as in the telephone number of Example 3), and novel error-handling subdialogues. Meanwhile, FORRSooth is already proving its robustness and habitability in preliminary trials.

Acknowledgements

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

References

Anderson, M. L., D. Josyula and D. Perlis 2003. Talking to computers. In Proc. of the Workshop on Mixed Initiative Intelligent Systems. In *Proc. of IJCAI Workshop on Mixed Initiative Intelligent Systems*.

Bangalore, S., P. Bouillier, A. Nasr, O. Rambow and B. Sagot 2009. MICA: a probabilistic dependency parser based on tree insertion grammars. . *Application Note. Human Language Technology and North American Chapter of the Association for Computational Linguistics*: 185-188.

Bohus, D. 2007. Error Awareness and Recovery in Conversational Spoken Language Interfaces. *Computer Science*. Pittsburgh, Carnegie Mellon University. Ph.D.

Bohus, D. and A. Raux 2009. The RavenClaw dialog management framework: Archictecture and systems. *Computers in Speech and Language* 23(3): 332-361.

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

Bohus, D. and A. I. Rudnicky 2003. RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. In *Proc. of Eurospeech 2003*.

Clark, H. H. and E. F. Schaefer 1989. Contributing to discourse. *Cognitive Science* 13: 259-294.

Cox, M. T. and A. Raja 2007. Metareasoning: A Manifesto, Technical Report. , BBN Technologies.

Epstein, S. L. 1994. For the Right Reasons: The FORR Architecture for Learning in a Skill Domain. *Cognitive Science* 18(3): 479-511.

Epstein, S. L. 1998. Pragmatic Navigation: Reactivity, Heuristics, and Search. *Artificial Intelligence* 100(1-2): 275-322.

Epstein, S. L. 2001. Learning to Play Expertly: A Tutorial on Hoyle. *Machines That Learn to Play Games*. Fürnkranz, J. and M. Kubat. Huntington, NY, Nova Science: 153-178.

Epstein, S. L., E. C. Freuder and R. J. Wallace 2005. Learning to Support Constraint Programmers. *Computational Intelligence* 21(4): 337-371.

Epstein, S. L. and S. Petrovic In press. Learning a Mixture of Search Heuristics. *Metareasoning: Thinking about thinking*, MIT Press.

Gagliolo, M. and J. Schmidhuber 2007. Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence* 47(3-4): 295-328.

Gomes, C. P. and B. Selman 2001. Algorithm portfolios. *Artificial Intelligence* 126(1-2): 43-62.

Gravano, A. 2009. Turn-Taking and Affirmative Cue Words in Task-Oriented Dialogue. *Department of Computer Science*. New York, Columbia University. Ph.D.

Gupta, N., G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi and M. Gilbert 2006. The AT&T spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing* 14(1): 213-222.

Levin, E., R. Pieraccini and W. Eckert 2000. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Trans. on Speech and Audio Processing* 8(1): 11-23.

Minton, S., J. A. Allen, S. Wolfe and A. Philpot 1995. An Overview of Learning in the Multi-TAC System. In *Proc. of First International Joint Workshop on Artificial Intelligence and Operations Research*, Timberline.

Nareyek, A. 2003. Choosing Search Heuristics by Non-stationary Reinforcement Learning. *Metaheuristics: Computer Decision-Making*. Resende, M. G. C. and J. P. deSousa. Boston, Kluwer: 523-544.

Passonneau, R., S. L. Epstein and J. B. Gordon 2009. Help Me Understand You: Addressing the Speech Recognition Bottleneck. In *Proc. of AAAI Spring Symposium on Agents that Learn from Human Teachers*, Palo Alto, CA, AAAI.

Passonneau, R. J., S. L. Epstein, T. Ligorio, J. Gordon and P. Bhutada 2010a. Learning about Voice Search for Spoken Dialogue. In *Proc. of NACL*.

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

Raux, A. and M. Eskenazi 2007. A Multi-layer architecture for semi-synchronous event-driven dialogue management. In *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2007)*, Kyoto.

Raux, A., B. Langner, A. W. Black and M. Eskenazi 2005. Let's Go Public! Taking a spoken dialog system to the real world. In *Proc. of Interspeech 2005 (Eurospeech)*, Lisbon.

Sacks, H., E. A. Schegloff and G. Jefferson 1974. A simplest systematics for the organization of turn-taking for conversation. *Language* 50(4): 696-735.

Streeter, M., D. Golovin and S. F. Smith 2007. Combining multiple heuristics online. In *Proc. of AAAI-07*, 1197-1203.

Walker, M. A., D. J. Littman, C. A. Kamm and A. Abella 1997. PARADISE: A framework for evaluation of spoken dialog agents. In *Proc. of 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain.

Williams, J. and S. Young 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems , vol. 21, no. 2, pp. 231–422,. *Computer Speech and Language* 21(2): 231-422.