CobotDS: A Spoken Dialogue System for Chat

Michael Kearns†, Charles Isbell‡, Satinder Singh§, Diane Litman¶, Jessica Howe♠

†Department of Computer and Information Science, University of Pennsylvania, mkearns@cis.upenn.edu
‡College of Computing, Georgia Institute of Technology, isbell@cc.gatech.edu
§Syntek Capital, baveja@cs.colorado.edu
¶Computer Science Department and LRDC, University of Pittsburgh, litman@cs.pitt.edu
♠ Laboratory for Artificial Intelligence, Massachussetts Institute of Technology, howej@ai.mit.edu

Abstract

We describe CobotDS, a spoken dialogue system providing access to a well-known internet chat server called LambdaMOO. CobotDS provides real-time, two-way, natural language communication between a phone user and the multiple users in the text environment. We describe a number of the challenging design issues we faced, and our use of summarization, social filtering and personalized grammars in tackling them. We report a number of empirical findings from a small user study.

Introduction

We describe the design, implementation and empirical experiences of CobotDS (for *Cobot Dialogue System*). CobotDS extends our ongoing work on Cobot (Isbell *et al.* 2000; 2001), a software agent that resides in a well-known, text-based internet chat environment called LambdaMOO. Founded in 1990, LambdaMOO (Cherny 1999; Foner 1997) is frequented by hundreds of users who converse with each other using both natural language text and *verbs* for expressing (in text) common real-world gestures (such as laughing, hugging, nodding and many others).[1] Cobot is one of the most popular LambdaMOO residents, and both chats with human users, and provides them with "social statistics" summarizing their usage of verbs and interactions with other users (such as who they interact with, who are the most "popular" users, and so on).

CobotDS provides LambdaMOO users with spoken telephony access to Cobot, and is an experiment in providing a rich social connection between a telephone user and the text-based LambdaMOO users. To support conversation, CobotDS passes messages and verbs from the phone user to LambdaMOO users (via automatic speech recognition, or ASR), and from LambdaMOO to the phone user (via text-to-speech, or TTS). CobotDS also provides "listening" (allowing phone users to hear a description of all LambdaMOO activity), chat summarization and filtering, personalized grammars, and many other features.

[1]Lines L1 and L14 in Table 1 illustrate the use of chat and verbs in LambdaMOO, respectively, and will be explained in detail below.

Our goal in building CobotDS was twofold. First, many LambdaMOO users log on as a form of regular social activity with a collection of friends and acquaintances. As a practical matter, we hope that CobotDS may provide an alternate means of access to LambdaMOO—either out of necessity (such as when a user is unable to access the internet), out of a desire to use a different input modality (speech instead of typing), or as an entertaining accompaniment to logging on directly. Second, we find it interesting to build a system that deliberately forms a connection, and blurs the distinction, between a world typically thought of as "real" (the world of telephones) and one typically thought of as "virtual". We also believe our experiences may hold lessons for future attempts to provide spoken access to text systems (such as instant messaging), and more generally for multimodal systems.

Traditional dialogue systems are designed to provide access to a relatively structured and static back-end database (such as airline reservation information), where users have well-defined, task-oriented goals (DARPA 2001; Sidner 2000). Compared to such systems, CobotDS is novel in a number of ways, and raises interesting challenges for dialogue system design.

First, CobotDS is one of the first dialogue systems to provide speech access to a complex social environment, where users participate primarily for entertainment or a sense of community. As such, it is difficult to anticipate user expectations or desires for CobotDS. For example, it was unclear whether users would prefer to use CobotDS for interaction with their LambdaMOO friends, or more as a passive listening mechanism. It was also unclear whether users would use the system primarily for LambdaMOO access when they were unable to log in to the text environment directly, or as an accompaniment to online access. Our approach to these questions is to try to provide enough functionality to shed some light on user needs, rather than fixing a model of them in advance. Thus, CobotDS should be viewed as an exploratory and evolving system.

Second, the "database" accessed by CobotDS is a dynamic and unstructured stream of natural language text, verb exchanges, and other actions. Conversational topics are interleaved, and users enter and exit at will. Providing the phone user with a useful view of this activity, as well as sufficiently rich means of participating in it, is a demanding

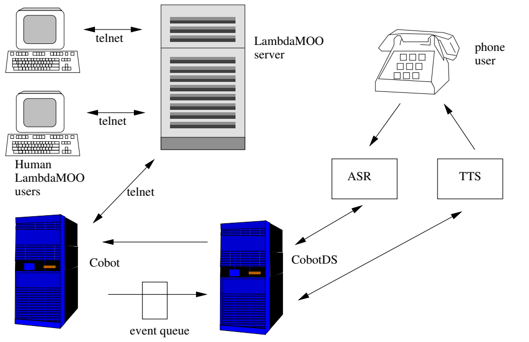

Figure 1: Architectural sketch of CobotDS, Cobot, and LambdaMOO, providing two-way communication between phone and chat users.

design problem. Some of the methods we applied and will discuss include summarization and filtering (to reduce the load on the phone user during busy times), and the use of "personal" grammars (to provide phone users with a rich set of personalized utterances within the current limitations of ASR).

Third, there are many issues of imbalance and asynchrony between the CobotDS phone user and the LambdaMOO text users. The phone user has a limited channel *to* LambdaMOO (due to the imperfections of ASR), and a potentially rich and cognitively challenging channel *from* LambdaMOO (as the phone user must absorb the action in real time via TTS, and does not have text-based scroll back capabilities). We have taken first steps in bridging this gap, and dealing with its consequences—such as time lag—but clearly more research is needed.

In the remainder of this paper, we sketch some salient points about the system architecture, provide an overview of the functionality provided by CobotDS to both phone and LambdaMOO users, and report on a number of interesting empirical findings derived from a small user study.

CobotDS Architecture

Cobot connects to the LambdaMOO server as a client, just as any human user would, and maintains information about LambdaMOO that he uses for his chat interactions and social statistics services. Although CobotDS and Cobot appear to users as a single system, in reality CobotDS is implemented as a separate set of processes handling ASR, TTS, telephony, dialogue management, and semantic processing, built using a general purpose platform for spoken dialogue systems (Levin *et al.* 1999). Thus, Cobot interacts with LambdaMOO and its users, while CobotDS interacts with the phone user. Figure 1 provides a sketch of the overall architecture.

LambdaMOO events or messages to be communicated from Cobot to CobotDS (and thus to the phone user) are handled by an *event queue* lying between Cobot and CobotDS. As such events occur, Cobot places them in the event queue, which CobotDS flushes at each dialogue turn. The event queue is a buffering mechanism designed to address the fact that the rate at which events can be pushed to the phone user may sometimes lag considerably behind the rate at which they are being generated in LambdaMOO. The desire to minimize this lag led to our implementation of some interesting queue *filtering* mechanisms (discussed later).

In the other direction, events to be passed from the phone user to LambdaMOO are immediately passed from CobotDS to Cobot. No explicit queuing mechanism is necessary. As Cobot notices messages from the phone user he processes them, just as he processes events from LambdaMOO as they occur.

CobotDS Functionality Overview

Table 1 is a sample dialogue illustrating all of the commands provided by CobotDS to both the phone and LambdaMOO users, and will be used extensively for expository purposes. Although hypothetical, the dialogue is representative of the actual dialogues discussed in the section on empirical findings, and all of the functionality shown is implemented.

After a brief login procedure (turns C1 and C2 of Table 1)[2], the phone user is placed in the *main command loop*, where he is repeatedly prompted for a CobotDS command (as in line C3S). User response is interpreted through ASR, and the resulting command executed. For example, after C3U, on the phone side, CobotDS tells the phone user what was heard, executes the command, and passes any messages or verbs directed towards them (via Cobot) from LambdaMOO users (line C4S). In LambdaMOO, completion of the CobotDS user login causes Cobot to announce the call in LambdaMOO (lines L3-L4).

Communicating with the Phone User

LambdaMOO users can pass messages to the phone user by directing a text message beginning with **phone:** to Cobot, as on line L7. If the text following **phone:** can be interpreted as a common LambdaMOO verb (line L11), it is passed on as such to the phone user via TTS (line C9S); otherwise, it is treated as an act of directed speech (lines L7 and C6S). In either case, the LambdaMOO user invoking **phone:** on Cobot is credited with the action.

Basic LambdaMOO Verbs

We now move on to discuss the commands provided to the phone user that can be invoked after prompting in the main command loop. The most basic functionality is the replication of the most common LambdaMOO verbs, such as waving and hugging (currently CobotDS supports 38 such verbs). These verbs can be invoked either without or with a user name (lines C4U and C6U, respectively), generating different behaviors by Cobot in LambdaMOO (lines L5 and L9-L10, respectively). Note that LambdaMOO users are already accustomed to using the same simple syntax for these verbs.

Conversation and Grammars

Despite the importance of verb use, LambdaMOO remains primarily a chat environment. To the extent possible, we felt

[2] To control system access and load, only a limited set of users have been granted CobotDS passwords. There are currently 33, including the authors.

CobotDS Transcript
(spoken; phone user and CobotDS alternate)

C1S. **Hello, who is this?** *(CobotDS prompt)*
C1U. *Buster. (phone user response)*
C2S. **Secret password?**
C2U. *Eight three two two.* [C(a), successful login]
C3S. **Hi, Buster. What do you want to do?**
C3U. *Who.*
C4S. **You who. I am with Cocobot, Betwixt, and HFh.**
C4U. *Wave.* [C(b)]
C5S. **You wave.**
C5U. *Where.*
C6S. **You where. I am in The Cobotatory. HFh says How was the movie?, to you.** [L(a)]
C6U. *Smile Betwixt.* [C(c)]
C7S. **You smile to Betwixt.**
C7U. *Say HFh.*
C8S. **What message would you like to pass?**
C8U. *Fantastic.* [C(d), smalltalk ASR grammar]
C9S. **You say Fantastic, to HFh. Betwixt waves to you.** [L(b)]
C9U. *Say.*
C10S. **What message would you like to pass?**
C10U. *What's going on in the MOO?.* [C(e), smalltalk ASR grammar]
C11S. **You say What's going on in the MOO, to everyone.**
C11U. *Summarize.*
C12S. **You summarize. Recently, HFh and Betwixt chatted. Betwixt and GreenBean bickered. HFh and Natto had a friendly conversation. I added to the noise. HFh made the most noise.**
C12U. *Grammar (ASR rejects the utterance, and returns nothing).*
C13S. **Next. HFh says I am providing hoops wisdom to Betwixt, to you, and then, Betwixt says, HFh is pontificating on the Knicks again. Where are you?, to you.** [L(c),L(d)]
C13U. *Grammar.*
C14S. **Which grammar would you like to use? The choices are smalltalk, personal, and cliche.**
C14U. *Personal.*
C15S. **The grammar is now set to Buster.**
C15U. *Say Betwixt.*
C16S. **What message would you like to pass?**
C16U. *I am in Hoboken.* [C(f), Buster personal grammar]
c17S. **You say I am in Hoboken, to Betwixt.**
C17U. *Listen.* [C(g), listen mode turned on]
C18S. **You listen. Betwixt gives you a nod. HFh to Betwixt, Sprewell can always be counted on to go straight to the rim, but Houston settles for jumpers from the parking lot, and then, I grin to HFh.** [L(e),L(f),L(g)]
C18U. *Grin HFh.* [C(h)]
C19S. **You grin to HFh. Betwixt to HFh, With Camby's rebounding they have a chance to at least win the East, and then, HFh to Betwixt, Ummm, as long as he doesn't send his coach to intensive care first, and then, Natto teleports in from the Living Room, and then, Betwixt to HFh, Good point.** [L(h),L(i),L(j),L(k)]

LambdaMOO Transcript
(text, joined in progress; multiple users and Cobot in ongoing, interleaved dialogues)

L1. HFh [to Betwixt]: And thanks to TiVo, I was able to see the game when I got home.
L2. Betwixt [to HFh]: I missed the first half because I was busy shoveling our record snowfall. But the second half was pretty spectacular.
L3. Cobot turns to pick up the phone. [C(a)]
L4. Cobot begins talking to Buster! [C(a)]
L5. Cobot holds up a sign: Buster passes on a wave from the phone. [C(b)]
L6. HFh pats Cobot on the head.
L7. HFh [to Cobot]: phone: How was the movie? [L(a)]
L8. Betwixt [to HFh]: Overall, it's one of their better stretches this year.
L9. Cobot smiles at Betwixt. [C(c)]
L10. Cobot [to Betwixt]: That was from Buster. [C(c)]
L11. Betwixt [to Cobot]: phone: wave [L(b)]
L12. Cobot [to HFh]: Fantastic [C(d)]
L13. Cobot [to HFh]: That was from Buster. [C(d)]
L14. HFh nods to Betwixt.
L15. HFh [to Betwixt]: But it can't last. The soap opera always takes over at some point. Drama over basketball.
L16. Cobot holds up a sign: Buster says, 'What's going on in the MOO' from the phone. [C(e)]
L17. HFh [to Cobot]: phone: I am providing hoops wisdom to Betwixt. [L(c)]
L18. Betwixt [to Cobot]: phone: HFh is pontificating on the Knicks again. Where are you? [L(d)]
L19. Betwixt [to HFh]: Your problem is you have no faith.
L20. HFh [to Betwixt]: Yours is that you have too much in the wrong team.
L21. HFh cackles madly at Betwixt!
L22. Cobot holds up a sign: Buster says, 'I am in Hoboken' from the phone. [C(f)]
L23. Betwixt [to Cobot]: phone: nod [L(e)]
L24. Cobot holds up a sign: Buster passes on a listen from the phone. [C(g)]
L25. HFh [to Betwixt]: Sprewell can always be counted on to go straight to the rim, but Houston settles for jumpers from the parking lot. [L(f)]
L26. Cobot [to HFh]: This is from Buster: [C(h),L(g)]
L27. Cobot grins at HFh. [C(h),L(g)]
L28. Betwixt [to HFh]: With Camby's rebounding they have a chance to at least win the East. [L(h)]
L29. HFh [to Betwixt]: Ummm, as long as he doesn't send his coach to intensive care first. [L(i)]
L30. Natto teleports in from the Living Room. [L(j)]
L31. Betwixt [to HFh]: Good point [L(k)].

Table 1: Parallel CobotDS-LambdaMOO transcripts for a sample dialogue. The left-hand column shows a dialogue with CobotDS, with the ith dialogue turn numbered CiS (system) and CiU (user). The right-hand side shows the activity taking place in LambdaMOO during this call, with the ith event numbered Li. Events in CobotDS that cause announcements in LamdbaMOO are labeled as [C(x)] on the left-hand side, with the same label marking the resulting announcements on the right-hand side. Similarly, events in LambdaMOO that cause announcements in CobotDS are labeled as [L(x)] on the right-hand side, with the same label marking the resulting announcements on the left-hand side.

it important to provide some reasonably expressive mechanisms for conversational exchange. It is rather easy to provide unconstrained message-passing from LambdaMOO to the phone user; however, communication from the phone user to LambdaMOO is severely hampered by the limitations of ASR. Nonetheless, CobotDS provides a **say** command (that takes an optional user name argument). Upon invoking the **say** command (as in line C7U), the phone user enters a one-step sub dialogue where he is prompted for the utterance he wishes to pass (turn C8). This utterance is then given to ASR, along with the recognition *grammar* that is currently in effect. Before passing the output of ASR on to LambdaMOO, CobotDS performs a *backmatching* step: the ASR output phrase is matched against each of the phrases in the current recognition grammar, and the phrase most similar to the ASR phrase is then passed on to LambdaMOO (lines L12-L13).[3]

CobotDS has two built-in grammars, the *smalltalk* and *cliche* grammars, and a *personal* grammar that differs for each phone user. The smalltalk grammar consists of 228 hand-constructed phrases providing basic conversational queries, responses, and remarks. (Examples include variants of "yes" and "no"; locational assertions such as "I am at home"; exclamations like "fantastic" or "terrible"; LambdaMOO-specific phrases such as "What's going on in the MOO?" and "My connection is down"; and conversational staples such as "How are you" and "I am fine".) The cliche grammar consists of 2950 common English sayings (such as "It takes one to know one" and "A rose by any other name would smell as sweet"). The personal grammar consists of a list of phrases provided by each phone user. The smalltalk grammar is initially in effect. The phone user can change the grammar by using the **grammar** command (line C13U), initiating a one-step sub dialogue where the phone user is prompted for a grammar name (turn C14).[4]

The hope is that the smalltalk grammar provides the rudiments of conversation, the cliche grammar contains common witticisms allowing the phone user to occasionally make an appropriate remark on LambdaMOO action, and the personal grammar can be used for the favorite phrases of the user. The use of multiple grammars for the **say** command, and allowing users to construct grammars, are potentially interesting experiments in placing users in more direct control of technology.

Listening and Social Filtering

In addition to allowing the phone user to send and receive messages, we anticipate that phone users may sometimes wish to hear *all* the action in LambdaMOO. Indeed, it is possible that some phone users may primarily use CobotDS as a passive source of entertainment (somewhat like listening to a radio talk show), interacting only minimally. The **listen** command (line C17U) puts CobotDS into a special mode where every system prompt in the main command loop includes a *filtered* version of all the activity taking place in LambdaMOO during the last dialogue turn (lines C18S and C19S). This mode is turned off by invoking the **mute** command.

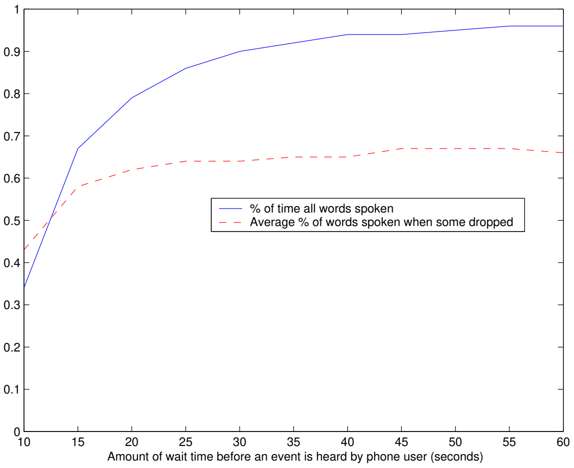

Figure 2: Information loss due to filtering, computed from LambdaMOO event logs. The x-axis shows the maximum number of seconds allotted between the time an event occurs in LambdaMOO and the time its filtered version is read to the phone user. (This interval includes the overhead time required to record the phone user's command, about 7 seconds.) For the solid line, the y-axis shows the fraction of turns for which no filtering of events was necessary (because in the allotted time, it was possible to read out completely the events generated in the last dialogue turn). For the dashed line, the y-axis shows the ratio of the length of the filtered text to the length of the original whenever filtering had to be used. This graph was computed from several days of LambdaMOO activity; nearly identical plots were obtained for other time periods. Note that increasing the allowed lag beyond 20-25 seconds (corresponding to about 59 words) does not significantly increase the overall information reaching the phone user.

One difficulty that arises in implementing **listen** is the empirical fact that the rate at which TTS can read text can lag considerably behind the rate at which text is being generated in LambdaMOO. Our solution is to *fix* the length of the text that will be read with each prompt (independent of how much activity took place), but to use this text to summarize *all* activity since the last turn; hence the need for a filtering mechanism.

A trade-off arises in choosing the length of the text to be read: a shorter length results in lower maximum time lag for the phone user, but forces more severe filtering. We empirically determined that a length of 59 words strikes a good balance (see Figure 2). We also decided to *personalize* the filtering for each user based on their past *social* interactions in the text environment, as observed by Cobot. There is an ordered set of filtering rules, and each rule is applied in turn until the desired length is met. While the earliest of these rules is not personalized, eventually all the activity generated by users with whom the phone user has little or no past

[3] Here, the similarity of two phrases is determined by treating each as a vector of word counts and computing the normalized inner product between them. This is a common and well-understood technique from information retrieval.

[4] The user only has to say "personal", because his identity, and thus the grammar to load, is known from the login process.

interaction may be dropped from the summary.

Special Informational Commands

Finally, CobotDS provides commands giving information on the current state of the text environment (which generate responses to phone users but not to LambdaMOO users). The **where** and **who** commands (lines C5U and C3U) tell the phone user which room of LambdaMOO Cobot currently occupies[5] (line C6S), and which LambdaMOO users are present there (line C4S), respectively.

More interesting is the **summarize** command (line C11U), which presents the phone user with a coarse summary of the last 10 minutes of activity in LambdaMOO (or the last n minutes, if the phone user utters an integer argument). Cobot uses his social statistics to compute which users have generated the most activity (in terms of number of verb invocations), which pairs of users have interacted the most, and which players have entered and exited (as in line C12S). The pairwise interactions may be characterized as "friendly" (if the fraction of verbs such as **smile** exceeds a certain threshold), or "nasty" (if the fraction of verbs such as **kick** exceeds a threshold).

The motivation behind **summarize** is to allow the phone user, either upon login or later in the dialogue, to get a brief synopsis of who is present and the nature of their interactions, if any. We view the command as a "batch" operation crudely summarizing a long time period. By contrast, the filtering mechanism used by the **listen** command gives a "play by play" synopsis of the action, and is designed for real-time summarization.

Empirical Findings

As of this writing, over 300 calls have been placed to CobotDS. Excluding the authors, 18 users have placed at least one call to the system. CobotDS is thus an active service provided to select LambdaMOO users, and a wide variety of experiences and usage patterns have been reported to and observed by us.

To quantify the empirical use of CobotDS in a more controlled manner, we conducted a small and informal user study during a recent two-week period. We restricted our attention to five (non-author) users who made at least 5 calls each to CobotDS during the study period (two females, three males, all native speakers). This section describes a number of quantitative and anecdotal findings derived from our complete logs (on both the CobotDS and LambdaMOO sides) of the calls made by this restricted user group, as well as from the personal grammars submitted by the users. Note, however, that the logs represent CobotDS's (post-ASR) interpretation of the user's utterances, and are thus only an approximation of the phone user's true intent.

Some of our findings are given in Table 2. Each column represents a user, and the rows show CobotDS or LambdaMOO usage statistics for that user. Note that there are many differences between users. For example, the average number of dialogue turns per call (row 2) varies from 31.6 for Benny to 106.6 for Nisa. This latter figure corresponds to an average call duration of over 20 minutes, suggesting that CobotDS is being used for extended social interaction by some users.

This view is supported by other figures in the table. Consider turns where (CobotDS believes that) the phone user executed an *interaction* command (either **say** or any LambdaMOO verb), affecting both the phone and LambdaMOO users. The fraction of interaction commands (row 4) across all users is 0.50 (ranging from Etoile's 0.33 to Huey's 0.59). Among the interaction turns, users vary in their use of **say** versus other verbs (row 5). Etoile is relatively verbal (at 0.83), while Huey is relatively gestural (at 0.38).

Users often provided username arguments for interaction commands (*directed* interaction, rows 6 and 7), and tended to interact with multiple LambdaMOO users (rows 8 and 9). Interestingly, the LambdaMOO users communicating with the phone user tended to use **say** more frequently (about 79 percent of the time) than the verbs (row 11). This is in contrast with the usage of **say** between LambdaMOO users (roughly 43 percent).

The **listen** command was also reasonably popular, with an average of 0.61 invocations per dialogue across all users (row 12). Some users enter **listen** mode and stay there for considerable periods of time (row 14, particularly Huey), but users also tend to turn listen mode on only later in the dialogue (row 15, where we see that listen mode is first invoked after 38 turns on average). This is consistent with our informal observation that there is typically a flurry of interaction between the phone and LambdaMOO users when Cobot first announces that a user is on the phone, thus providing the phone user with a sense of participation, but that as this initial exchange dies down, they switch to listen mode. There is also some evidence for the use of **listen** in "radio" mode, in which the phone user simply listens to LambdaMOO without giving any CobotDS commands. For example, rows 16 and 17 demonstrate that user Nisa is silent (empty string returned from ASR) a much greater fraction of time in listen mode than in non-listen mode.

Recall that if the phone user is listening during a busy time, he may receive a filtered version of LambdaMOO activity (personalized by past social interaction). We note that our filtering mechanism performed reasonably well: the average real-time lag in hearing the filtered version of LambdaMOO events was only about 10 seconds (which includes the "overhead" time for recording the phone user's command). When filtering was necessary at all (that is, the filtered version of LambdaMOO events differed from the original), the filtered text length was on average 0.70 times the unfiltered length. This is quite consistent with Figure 2.

We also see that users took advantage of the **grammar** command: it was invoked an average of 1.4 times per dialogue across all users (row 18), and used to set the grammar to personal 69 percent of the time (row 19). The first change of grammar occurred relatively early (row 20), confirming our observation that users would often change to personal grammars quickly, and remain there for entire dialogue.

[5] LambdaMOO actually consists of multiple distinct chat rooms, connected by virtual exits and entrances that are navigable by standard system commands. Cobot divides his time between two different LambdaMOO rooms.

		Jethromeo	Etoile	Nisa	Huey	Benny	*Average*
1	number of dialogues	5	7	9	5	5	6.2
2	mean number of turns per dialogue	36.2	39.3	106.6	92.2	31.6	65.3
3	mean interactions to the MOO	8.2	6.85	29.1	27.2	5.8	16.0
4	fraction of interaction commands	0.43	0.33	0.55	0.59	0.36	0.50
5	fraction of say interactions	0.68	0.83	0.54	0.38	0.59	0.54
6	fraction of directed says	0.42	0.86	0.41	0.62	0.47	0.52
7	fraction of directed interactions	0.41	0.86	0.48	0.65	0.55	0.56
8	mean MOO recipients	2.6	3.0	5	5	2	3.7
9	mean MOO pushers	2	2.43	3.2	2.8	0.8	2.4
10	mean interactions from MOO	5.2	5.1	11.3	12.4	1.2	7.4
11	fraction of MOO say interactions	0.85	0.83	0.73	0.85	0.66	0.79
12	mean number of listens	0.6	0.57	0.44	1.2	0.4	0.61
13	mean number of mutes	0	0.43	0.71	0.8	0.2	0.42
14	mean turns taken for a listen	15	4.6	22.5	32.8	21	18.7
15	mean turn number to begin listen	63	30.5	43	39.7	15.5	38.4
16	fraction of silent turns in listen	0.53	0.5	0.73	0.46	0.19	0.52
17	fraction of silent turns in mute	0.46	0.48	0.48	0.52	0.59	0.49
18	mean grammar changes	0.8	1.5	1.5	1.6	1.0	1.4
19	fraction of grammar changes to personal	1.0	0.55	0.57	0.75	1.0	0.69
20	mean turn number of grammar change	8.5	17	6.2	9.75	7.2	9.8

Table 2: CobotDS usage statistics. Here we list statistics summarizing how CobotDS was used by our most active phone users during the recent two-week study. A *command* is any completed transaction with the system, from the (post-ASR) perspective of CobotDS. An *interaction* is a command that results in some kind of verbal or gestural communication with a LambdaMOO user (either the **say** command or a LambdaMOO verb). A **say** or other interaction is called *directed* if it addresses a specific LamdbaMOO user. A user is always considered to be in either *listen* or *mute* mode. Finally, a *silent turn* happens when a user decides not to issue any kind of command (or when ASR misrecognizes a command as silence). See the text for a more complete discussion.

Personal grammars had an average length of 29.6 phrases (ranging from 6 to 60). Typically a personal grammar included set phrases commonly invoked by the user in LambdaMOO itself. Thus Huey's personal grammar contained "Can I get an amen", while Nisa included "Chillin' like a villain". Interestingly, some personal grammars evolved to include phrases compensating for ASR errors and the limitations of the grammars themselves.[6] Thus, users added phrases such as "I can't answer that question given my limited vocabulary", "I don't have the words to convey my feeling towards the issue at hand", "I didn't mean to say that", and "Cobot misunderstood that's not what I meant". Some users added sentences containing acoustically distinct keywords to increase the chances of recognition, sometimes pronouncing just those keywords (taking advantage of our use of backmatching). Users also included phrases along the lines of the smalltalk grammar, but that were missing, such as "Going to lunch." By contrast, users made minimal use of the cliche grammar.

Users in the study also completed a brief survey that queried their experiences with CobotDS and solicited suggestions for improvements to the system. Overall, users seemed to find CobotDS interesting, fun, and useful. However, common themes in the responses included frustration with poor ASR performance (particularly for the **say** command), and the difficulty of sustaining conversation with the restrictive grammars. Interesting suggestions included providing the ability to update the personal grammar instantaneously (which is unfortunately precluded by our internal development environment, but is available on some commercial platforms), and providing confirmation for the utterance in the **say** command. Initially, we believed that confirming long utterances would prove irritating, but are now contemplating allowing users the ability to turn it on and off for the **say** command. There were also comments suggesting that users be able to selectively **listen** to the activity generated by some, but not all, LambdaMOO users. This confirms our suspicion that the amount of LambdaMOO activity at busy times overwhelms the TTS-bound phone user, and that filtering activity by prior social interaction is an avenue worth exploring even further.

[6]During the trial, we gave our users the opportunity to occasionally update their personal grammars.

References

Cherny, L. 1999. *Conversation and Community: Discourse in a Social MUD.* CLFI Publications.

DARPA. 2001. Communicator web page, http://www.darpa.mil/ito/research/com/index.html.

Foner, L. 1997. Entertaining Agents: a Sociological Case Study. In *Proc. of Autonomous Agents.*

Isbell, C. L.; Kearns, M.; Kormann, D.; Singh, S.; and Stone, P. 2000. Cobot in LambdaMOO: A Social Statistics Agent. *Proc. of AAAI 2000.*

Isbell, C. L.; Shelton, C. R.; Kearns, M.; Singh, S.; and Stone, P. 2001. A Social Reinforcement Learning Agent. *Proc. of Autonomous Agents 2001.*

Levin, E.; Pieraccini, R.; Eckert, W.; Fabbrizio, G. D.; and Narayanan, S. 1999. Spoken language dialogue: From theory to practice. In *Proc. IEEE ASRU99 Workshop.*

Sidner, C., ed. 2000. *ANLP/NAACL Workshop on Conversational Systems.*