Using Reinforcement Learning to Create Communication Channel Management Strategies for Diverse Users

Rebecca Lunsford
Center for Spoken Lang. Understanding
Oregon Health & Science University
Beaverton, OR, USA
lunsforr@ohsu.edu

Peter Heeman
Center for Spoken Lang. Understanding
Oregon Health & Science University
Beaverton, OR, USA
heemanp@ohsu.edu

Abstract

Spoken dialogue systems typically do not manage the communication channel, instead using fixed values for such features as the amplitude and speaking rate. Yet, the quality of a dialogue can be compromised if the user has difficulty understanding the system. In this proof-of-concept research, we explore using reinforcement learning (RL) to create policies that manage the communication channel to meet the needs of diverse users. Towards this end, we first formalize a preliminary communication channel model, in which users provide explicit feedback regarding issues with the communication channel, and the system implicitly alters its amplitude to accommodate the user's optimal volume. Second, we explore whether RL is an appropriate tool for creating communication channel management strategies, comparing two different hand-crafted policies to policies trained using both a dialogue-length and a novel *annoyance* cost. The learned policies performed better than hand-crafted policies, with those trained using the annoyance cost learning an equitable tradeoff between users with differing needs and also learning to balance finding a user's optimal amplitude against dialogue-length. These results suggest that RL can be used to create effective communication channel management policies for diverse users.

Index Terms: communication channel, spoken dialogue systems, reinforcement learning, amplitude, diverse users

1 Introduction

Both Spoken Dialog Systems (SDS) and Assistive Technology (AT) tend to have a narrow focus, supporting only a subset of the population. SDS typically aim to support the "average man", ignoring wide variations in potential users' ability to hear and understand the system. AT aims to support people with a recognized disability, but doesn't support those whose impairment is not severe enough to warrant the available devices or services, or those who are unaware or have not acknowledged that they need assistance. However, SDS should be able to meet the needs of users whose abilities fall within, and between, the extremes of severly impaired and perfectly abled.

When aiming to support users with widely differing abilities, the cause of a user's difficulty is less important than adapting the communication channel in a manner that aids understanding. For example, speech that is presented more loudly and slowly can help a hearing-impaired elderly person understand the system, and can also help a person with no hearing loss who is driving in a noisy car. Although one user's difficulty is due to impairment and the other due to an adverse environment, a similar adaptation may be appropriate to both.

During human-human communication, speakers manage the communication channel; implicitly altering their manner of speech to increase the likelihood of being understood while concurrently economizing effort (Lindblom, 1990). In addition to these implicit actions, speakers also make statements referring to breakdowns in the communication channel, explicitly pointing out potential problems or corrections, (e.g. "Could you please speak up?") (Jurafsky et al., 1997).

As for human-computer dialogue, SDS are prone to misrecognition of users' spoken utterances. Much research has focused on developing techniques for overcoming or avoiding system misunderstandings. Yet, as the quality of automatic speech recognition improves and SDS are deployed to diverse populations and in varied environments, systems will need to better attend to possible *human* misunderstandings. Future SDS will need to manage the communication channel, in addition to managing the task, to aid in avoiding these misunderstandings.

Researchers have explored the use of reinforcement learning (RL) to create dialogue policies that balance and optimize measures of task success (e.g., see (Scheffler and Young, 2002; Levin et al., 2000; Henderson et al., 2008; Walker, 2000)). Along these lines, RL is potentially well suited to creating policies for the subtask of managing the communication channel, as it can learn to adapt to the user while continuing the dialogue. In doing so, RL may choose actions that appear costly at the time, but lead to better overall dialogues.

Our long term goal is to learn how to manage the communication channel along with the task, moving away from just "what" to say and also focusing on "how" to say it. For this proof-of-concept, our goals are twofold: 1) to formalize a communication channel model that encompasses diverse users, initially focusing just on explicit user actions and implicit system actions, and 2) to determine whether RL is an appropriate tool for learning an effective communication channel management strategy for diverse users. To explore the above issues, we use a simple communication channel model in which the system needs to determine and maintain an amplitude level that is pleasant and effective for users with differing amplitude preferences and needs. As our goal includes decreasing the amount of potentially annoying utterances (i.e., those in which the system's amplitude setting is in discord with the user's optimal amplitude), we introduce a user-centric cost metric, which we have termed *annoyance cost*. We then compare hand-crafted policies against policies trained using both annoyance and more traditional dialogue-length cost components.

2 Related Work

2.1 How People Manage the Channel

When conversing, speakers implicitly adjust features of their speech (e.g., speaking rate, loudness) to maintain the communication channel. For example, speakers produce Lombard speech when in noisy conditions, produce clear speech to better accommodate a hard of hearing listener, and alter their speech to more closely resemble the interlocutor's (Junqua, 1993; Lindblom, 1990). These changes increase the intelligibility of the speech, thus helping to maintain the communication channel (Payton et al., 1994). Research has also shown that speakers adjust their speaking style when addressing a computer; exhibiting the same speech adaptations seen during human-human communication (Bell et al., 2003; Lunsford et al., 2006).

In addition to altering their speech implicitly, speakers also explicitly point out communication channel problems (Jurafsky et al., 1997). Examples include; requesting a change in speaking rate or amplitude ("Could you please speak up?"), explaining sources of communication channel interference ("Oh, that noise is the coffee grinder."), or asking their interlocutor to repeat an utterance ("What was that?"). These explicit utterances identify some issue with the communication channel that must be remedied before continuing the dialogue. In response, interlocutors will rewind to a previous point in the dialogue and alter their speech to ensure they are understood. This approach, of adapting ones speech in response to a communication problem, occurs even when conversing with a computer (Stent et al., 2008).

Both implicit speech alterations and explicit utterances regarding the communication channel often address issues of amplitude. This is to be expected, as speaking at an appropriate amplitude is critical to maintaining an effective communication channel, with sub-optimal amplitude affecting listeners' understanding and performance (Baldwin and Struckman-Johnson, 2002). In addition, Baldwin (2001) found that audible, but lowered, amplitude can negatively affect both younger and older subjects' reaction time and ability to respond correctly while multitasking, and that elderly listeners are likely to need higher amplitudes than younger listeners to maintain similar performance. Just as low amplitude can be difficult to understand, high amplitude can be annoying, and, in the extreme, cause pain.

2.2 How Systems Manage the Channel

Towards improving listener understanding in a potentially noisy environment, Martinson and Brock (2007) take advantage of the mobility and sensory capabilities of a robot. To determine the best course of action, the robot maintains a noise map of the environment, measuring the environmental noise prior to each TTS utterance. The robot then rotates toward the listener, changes location, alters its amplitude, or pauses until the noise abates. A similar technique, useful for remote listeners who may be in a noisy environment or using a noisy communication medium, could analyze the signal-to-noise ratio to ascertain the noise level in the listener's environment. Although these techniques may be useful for adjusting amplitude to compensate for noise in the listener's environment, they do not address speech alterations needed to accommodate users with different hearing abilities or preferences.

Given the need to adapt to individual users, it seems reasonable that users themselves would simply adjust volume on their local device. However, there are issues with this approach. First, manual adjustment of the volume would prove problematic when the user's hands and eyes are busy, such as when driving a car. Second, during an ongoing dialogue speakers tend to minimize pauses, responding quickly when given the turn (Sacks et al., 1974). Stopping to alter the amplitude could result in longer than natural pauses, which systems often respond to with increasingly lengthy 'time-out' responses (Kotelly, 2003), or repeating the same prompt endlessly (Villing et al., 2008). Third, although we focus on amplitude adaptations in this paper, amplitude is only one aspect of the communication channel. A fully functional communication channel management solution would also incorporate adaptations of features such as speaking rate, pausing, pitch range, emphasis, etc. This extended set of features, because of their number and interaction between them, do not readily lend themselves to listener manipulation.

3 Reinforcement Learning

RL has been used to create dialogue strategies that specify what action to perform in each possible system state so that a minimum dialogue cost is achieved (Walker, 2000; Levin et al., 2000). To accomplish this, RL starts with a policy, namely what action to perform in each state. It then uses this policy, with some exploration, to estimate the cost of getting from each state with each possible action to the final state. As more simulations are run, RL refines its estimates and its current policy. RL will converge to an optimal solution as long as assumptions about costs and state transitions are met. RL is particularly well suited for learning dialogue strategies as it will balance opposing goals (e.g., minimizing excessive confirmations vs. ensuring accurate information).

RL has been applied to a number of dialogue scenarios. For form-filling dialogues, in which the user provides parameters for a database query, researchers have used RL to decide what order to use when prompting for the parameters and to decrease resource costs such as database access (Levin et al., 2000; Scheffler and Young, 2002). System misunderstanding caused by speech recognition errors has also been modeled to determine whether, and how, the system should confirm information (Scheffler and Young, 2002). However, there is no known work on using RL to manage the communication channel so as to avoid *user* misunderstanding.

User Simulation: To train a dialogue strategy using RL, some method must be chosen to emulate realistic user responses to system actions. Training with actual users is generally considered untenable since RL can require millions of runs. As such, researchers create simulated users that mimic the responses of real users. The approach employed to create these users varies between researchers; ranging from simulations that employ only real user data (Henderson et al., 2008), to those that model users with probabilistic simulations based on known realistic user behaviors (Levin et al., 2000). Ai et al. suggest that less realistic user simulations that allow RL to explore more of the dialogue state space may perform as well or better than simulations that statistically recreate realistic user behavior (Ai et al., 2007). For this proof-of-concept work, we employ a hand-crafted user simulation that allows full exploration of the state space.

Costs: Although it is agreed that RL is a viable approach to creating optimal dialogue policies, there remains much debate as to what cost functions result in the most useful policies. Typically, these costs include a measure of efficiency (e.g., number of turns) and a measure of solution quality (e.g., the user successfully completed the transaction) (Scheffler and Young, 2002; Levin et al., 2000). For managing the communication channel, it is unclear how the cost function should be structured. In this work we compare two cost components, a more traditional dialogue-length cost versus a novel annoyance cost, to determine which best supports the creation of useful policies.

4 Communication Channel Model

Based on the literature reviewed in Section 2.1, we devised a preliminary model that captures essential elements of how users manage the communication channel. For now, we only include explicit user actions, in which users directly address issues with the communication channel, as noted by Jurafsky et al. (1997). In addition, the users modeled are both consistent and amenable; they provide feedback every time the system's utterances are too loud or too soft, and abandon the interaction only when the system persists in presenting utterances outside the user's tolerance (either ten utterances that are too loud or ten that are too soft).

For this work, we wish to create policies that treat all users equitably. That is, we do not want to train polices that give preferential treatment to a subset of users simply because they are more common. To accomplish this, we use a flat rather than normal distribution of users within the simulation, with both the optimal amplitude and the tolerance range randomly generated for each user. To represent users with differing amplitude needs, simulated users are modeled to have an optimal amplitude between 2 and 8, and a tolerance range of 1, 3 or 5. For example, a user may have a optimal amplitude of 4, but be able to tolerate an amplitude between 2 and 6.

When interacting with the computer, the user responds with: (a) the answer to the system's query if the amplitude is within their tolerance range; (b) too soft (TS) if below their range; or (c) too loud (TL) if the amplitude is above their tolerance range. As a simplifying assumption, TS and TL represent any user responses that address communication channel issues related to amplitude. For example, the user response "Pardon me?" would be represented by TS and "There's no need to shout!" by TL. With this user model, the user only responds to the domain task when the system employs an amplitude setting within the user's tolerance range.

For the system, we need to ensure that the system's amplitude range can accommodate any user-tolerable amplitude. For this reason, the system's amplitude can vary between 0 and 10, and is initially set to 5 prior to each dialogue. In addition to performing domain actions, the system specifies the amount the amplitude should change: -2, -1, +0, +1, +2. Each system communication to the user consists of both a domain action and the system's amplitude change. Thus, the system manages the communication channel using only implicit actions. If the user responds with TS or TL, the system will then restate what it just said, perhaps altering the amplitude prior to re-addressing the user.

5 Hand-crafted Policies

To help in determining whether RL is an appropriate tool for learning communication channel management strategies, we designed two hand-crafted policies for comparison. The first handcrafted policy, termed *no-complaints*, finds a tolerable amplitude as quickly as possible, then holds that amplitude for the remainder of the dialogue. As such, this policy only changes the amplitude in response to explicit complaints from the user. Specifically, the policy increases the amplitude by 2 after a TS response, and drops it by 2 after a TL. If altering the amplitude by 2 would cause the system to return to a setting already identified as too soft or too loud, the system uses an amplitude change of 1.

The second policy, termed *find-optimal*, searches for the user's optimal amplitude, then maintains that amplitude for the remainder of the dialogue. For this policy, the system first increases the amplitude by 1 until the user responds with TL (potentially in response to the system's first utterance), then decreases the amplitude by 1 until the user either responds with TS or the optimal amplitude is clearly identified based on the previous feedback. An amplitude change of 2 is used only when both the optimal amplitude is obvious and a change of 2 will bring the amplitude setting to the optimal amplitude.

6 RL and System Encoding

To learn communication channel management policies we use RL with system and user actions specified using Information State Update rules (Henderson et al., 2008). Following Heeman (2007), we encode commonsense preconditions rather than trying to learn them, and only use a subset of the information state for RL.

Domain Task: We use a domain task that requires the user to supply 9 pieces of information, excluding user feedback relating to the communication channel. The system has a deterministic way of selecting its actions, thus no learning is needed for the domain task.

State Variables: For RL, each state is represented by two variables; *AmpHistory* and *Progress*. *AmpHistory* models the user by tracking all previous user feedback. In addition, it tracks the current amplitude setting. The string contains one slot for each potential amplitude setting (0 through 10), with the current setting contained within "[]". Thus, at the beginning of each interaction, the string is "`-----[-]-----`", where "-" represents no known data. Each time the user responds, the string is updated to reflect which amplitude settings are too soft ("<"), too loud (">"), or within the user's tolerance ("O"). When the user responds with TL/TS, the system also updates all settings above/below the current setting. The *Progress* variable is required to satisfy the Markov property needed for RL. This variable counts the number of successful information exchanges (i.e., the user did not respond with TS or TL). As the domain task requires 9 pieces of information, the Progress variable ranged from 1 to 9.

Costs: Our user model only allows up to 10 utterances that are too soft or too loud. If the cutoff is reached, the domain task has not been completed, so a solution quality cost of 100 is incurred. Cutting off dialogues in this way has the additional benefit of preventing a policy from looping forever during testing. During training, to allow the system to better model the cost of choosing the same action repeatedly, we use a longer cutoff of 1000 utterances rather than 10.

In addition to solution quality, two different cost components are utilized. The first, a dialogue-length cost (DC), assigns a cost of 1 for each user utterance. The second, an annoyance cost (AC), assigns a cost calculated as the difference between the system's amplitude setting and the user's optimal amplitude. This difference is multiplied by 3 when the system's amplitude setting is below the user's optimal. This multiplier was chosen based on research that demonstrated increased response times and errors during cognitively challenging tasks when speech was presented below, rather than above, typical conversational levels (Baldwin and Struckman-Johnson, 2002). Thus, only utterances at the optimal amplitude have no cost.

7 Results

With the above system and user models, we trained policies using the two cost functions discussed above, eight with the DC component and eight using the AC component. All used Q-Learning and the ϵ-greedy method to explore the state space with ϵ set at 20% (Sutton and Barto, 1998). Dialogue runs were grouped into epochs of 100; after each epoch, the current dialogue policy was updated. We trained each policy for 60,000 epochs. After certain epochs, we tested the policy on 5000 user tasks.

For our simple domain, the solution quality cost remained 0 after about the 100th epoch, as all policies learned to avoid user abandonment. Because of this, only the dialogue-length cost(DC) and annoyance cost(AC) components are reflected in the following analyses.

7.1 DC-Trained Policies

By 40,000 epochs, all eight DC policies converged to one common optimal policy. Dialogues resulting from the DC policies average 9.76 user utterances long. DC policies start each dialogue using the default amplitude setting of 5. After receiving the initial user response, they aggressively explore the amplitude range. If the initial user response is TL (or

DC			
AmpHistory	System	Amp	User
-----[-]-----	Query$_1$ +0	5	TS
<<<<<[<]-----	Query$_1$ +2	7	Answer
<<<<<<-[0]---	Query$_2$ +0	7	Answer
<<<<<<-[0]---	Query$_3$ +0	7	Answer
<<<<<<-[0]---	Query$_4$ +0	7	Answer
<<<<<<-[0]---	Query$_5$ +0	7	Answer
<<<<<<-[0]---	Query$_6$ +0	7	Answer
...	...	...	...
dialogue length cost = 10			

AC			
AmpHistory	System	Amp	User
-----[-]-----	Query$_1$ +1	6	TS
<<<<<<[<]----	Query$_1$ +1	7	Answer
<<<<<<<[0]---	Query$_2$ +1	8	Answer
<<<<<<<0[0]--	Query$_3$ +1	9	Answer
<<<<<<00[0]-	Query$_4$ +1	10	TL
<<<<<<<000[>]	Query$_4$ -2	8	Answer
<<<<<<<0[0]0>	Query$_5$ +0	8	Answer
...	...	...	...
annoyance cost = 12			

Table 1: Comparison of DC (left) and AC (right) interactions with a user who has an optimal amplitude of 8 and a tolerance range of 3. The policies continue as shown, without changing the amplitude level, until all 9 queries are answered.

TS), they continue by decreasing (or increasing) the amplitude by -2 (or +2) until they find a tolerable volume, in which case they stop. Table 1 illustrates the above noted aspects of the policy. Additionally, if the policy receives user feedback that is contrary to the last feedback (i.e., TS after TL, or TL after TS), the policy backtracks one amplitude setting. In addition, if the current amplitude is near the boundary (3 or 7), the policy will change the volume by -1 or +1 as changing it by -2 or +2 would cause it to move outside users' amplitude range of 2-8. In essence, the DC policies are quite straightforward; aggressively changing the amplitude if the user complains, and assuming the amplitude is correct if the user does not complain.

7.2 AC-Trained Policies

By 55,000 epochs, AC policies converged to one of two optimal solutions, with an average annoyance cost of 7.49. As illustrated in Table 1, the behavior of the AC policies is substantially more complex than the DC policies. First, the AC policies start by increasing the amplitude, delivering the first utterance at a setting of 6 or 7. Second, the policies do not stop exploring after they find a tolerable setting, instead attempting to bracket the user's tolerance range, thus identifying the user's optimal amplitude. Third, AC policies sometimes avoid lowering the amplitude, even when doing so would concretely identify the user's optimal amplitude. By doing so, the policies potentially incur a cost of 1 for all following turns, but avoid incurring a one time cost of 3 or 6. In essence, the AC policies attempt to find the user's optimal amplitude but may stop short as they approach the end of the dialogue, favoring a slightly too high amplitude over one that might be too low.

7.3 Comparing AC- and DC- Trained Policies

The costs for the AC and DC trained policy sets cannot be directly compared as each set used a different cost function. However, we can compare them using each others' cost function.

First, we compare the two sets of policies in terms of average dialogue-length. For example, in Table 1, following a DC policy results in a dialogue-length of 10. However, for the same user, following the AC policy results in a dialogue-length of 11, one utterance longer due to the TL response to Query$_4$.

The average dialogue-length of the DC and AC policies, averaged across users, is shown in the right-most two columns of Figure 1. As expected, the DC policies perform better in terms of dialogue-length, averaging 9.76 utterances long. However, the AC policies average 10.32 utterances long, only 0.52 utterances longer. This similarity in length is to be expected, as system communication outside the user's tolerance range impedes progress and is costly using either cost component.

We also compared the AC and DC policies' average dialogue-length for users with the same optimal amplitude (i.e., each column shows the average cost across users with tolerance ranges of 1, 3 and 5), as shown in Figure 1. From this figure it is clear that there is little difference in dialogue-length between AC and DC policies for users with the same optimal amplitude. In addition, for both policies, the lengths are similar between users with differing optimal amplitudes.

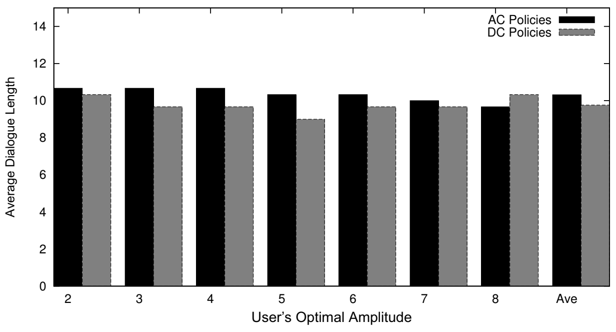

Figure 1: Comparison of the dialogue-length between AC and DC policies for users with differing optimal amplitudes.

Second, we compare the two sets of polices in terms of annoyance costs. For example, in Table 1, following the AC policy results in an annoyance cost of 12. For the same user, following the DC policy results in an annoyance cost of 36; 9 for Query$_1$ as it is three below the user's optimal amplitude, and 3 for each of the following nine utterances as they are all one below optimal.

As shown in the rightmost columns of Figure 2, DC policies average annoyance cost was 13.35, a substantial 78% increase over the average cost of 7.49 for AC policies. Figure 2 also illustrates that the AC and DC policies perform quite differently for users with differing optimal amplitudes. For example, users of the DC policies whose optimal is at (5), or slightly below (4), the system's default setting (5) average lower annoyance costs than those using the AC policies. However, these lowered costs for users in the mid-range is gained at the expense of users whose optimal amplitude is farther afield, especially those users requiring higher amplitude settings. This substantial difference between users with different optimal amplitudes is because, for DC policies, the interaction is often conducted at the very edge of the users' tolerance. In contrast, the AC policies risk more intolerable utterances, but use this information to decrease overall costs by better meeting users' amplitude needs. As such, users of the AC policies can expect the majority of the task to be conducted at, or only one setting above, their optimal amplitude.

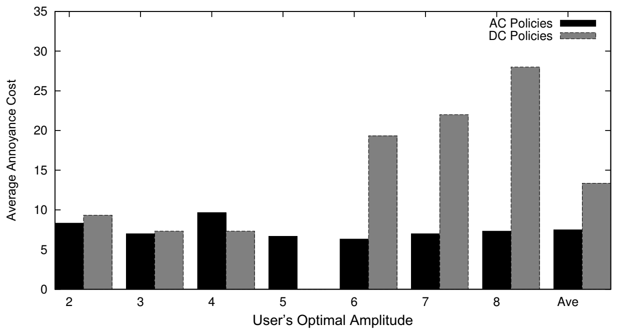

Figure 2: Comparison of the annoyance cost between AC and DC policies for users with differing optimal amplitudes.

7.4 Comparing Hand-crafted and Learned Policies

Each of the two hand-crafted policies were run with each user simulation (i.e., optimal amplitude from 2-8 and tolerance ranges of 1, 3, or 5). In addition, we varied the domain task size, requiring between 4 and 10 pieces of information. DC and AC policies were also trained for these domain task sizes.

As shown in Figure 3, The no-complain policy's annoyance costs ranged from 7.81 for dialogues requiring four pieces of information to 14.67 for those requiring ten pieces. The cost increases linearly with the amount of information required, because the no-complain policy maintains the first amplitude setting found that does not result in a user response of TS or TL. This ensures the amplitude setting is tolerable to the user, but may not be the user's optimal amplitude.

In contrast, the find-optimal policy's annoyance costs initially increase from 9.67 for four pieces of information to 12.24 for seven through ten pieces. The cost does not continue to increase when the amount of information required is greater than seven because, for dialogues long enough to allow the system to concretely identify the user's optimal amplitude, the cost is zero for all subsequent utterances.

Figure 3 also includes the mean annoyance cost for the DC and AC policies. Although one might expect the DC trained policies to resemble the no-complain policy, the learned policy performs slightly better. This difference is because the DC policies learn the range of users' optimal amplitude settings (2-8), and do not move the amplitude below 2 or above 8. In contrast, the no-complain policies

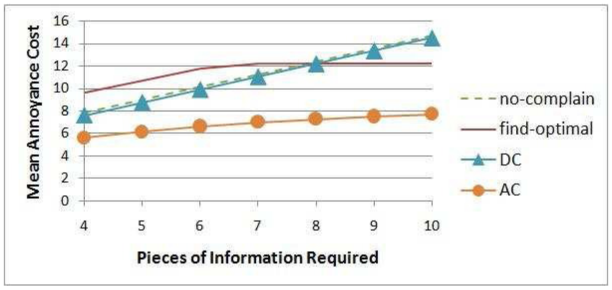

Figure 3: Average user annoyance costs for hand-crafted, DC and AC policies across dialogues requiring differing amounts of information.

behave consistently regardless of the current setting, and thus will incur costs for exploring settings outside the range of users' optimal amplitudes. Similarly, AC policies could be anticipated to closely resemble the find-optimal policy. However, the AC policies average cost is lower than the costs for either hand-crafted policy, regardless of the amount of information required. This difference is, in part, due to differences in behavior at the ends of the users' optimal amplitude range, like the DC policies. However, additional factors include the AC policies' more varied use of amplitude changes and their balancing of the remaining duration of the dialogue against the cost to perform additional exploration, as discussed in subsection 7.2.

8 Discussion and Future Work

The first objective of this work was to create a model of the communication channel that takes into account the abilities and preferences of diverse users. In this model, each user has an optimal amplitude, but will answer a system query delivered within a range around that amplitude, although they find non-preferred, especially too soft, amplitudes annoying. When outside the user's tolerance, the user provides explicit feedback regarding the communication channel breakdown. For the system, the model specifies a composite system action, pairing a domain action with a possible communication channel management action to change the amplitude. By modeling explicit user actions, and implicit system actions, this model captures some essential elements of how people manage the communication channel.

The second objective was to determine whether RL is appropriate for learning communication channel management. As expected, the learned policies found and maintained a tolerable amplitude setting and eliminated user abandonment. We also compared the learned policies with handcrafted solutions, and found that the learned policies performed better. This is primarily due to RL's ability to automatically balance the opposing goals of finding the user's optimal amplitude and minimizing dialogue-length.

An added benefit of RL is that it optimizes the system's behavior for the users on which it is trained. In this work, we purposely used a flat distribution of users, which caused RL to find a policy (especially when using annoyance costs) that does not penalize the outliers, which are usually those with special needs. In fact, we could modify the user distribution, or the simulated users' behavior, and RL would optimize the system's behavior automatically.

In this work, we contrasted dialogue length (DC) against annoyance cost (AC) components. We found that the AC and DC policies share the objective of finding an amplitude setting within the user's tolerance range because both incur stepwise costs for intolerable utterances. But, AC policies further refine this objective by incurring costs for tolerable, but non-optimal, amplitudes as well. AC policies are using information that is not explicitly communicated to the system, but which none-the-less RL can use while learning a policy.

As this was exploratory work, the user model does not yet fully reflect expected user behavior. For example, as the system's amplitude decreases, users may misunderstand the system's query or fail to respond at all. In future work we will use an enhanced user model that includes more natural user behavior. In addition, because we wanted the system to focus on learning a communication channel management strategy, the domain task was fixed. In future work, we will use RL to learn policies that both accomplish a more complex domain task, and model connections between domain tasks and communication channel management. Ultimately, we need to conduct user-testing to measure the efficacy of the communication channel management policies. We feel confident that learned policies trained using a communication channel model which reflects the range of users' abilities and preferences will prove effective for supporting all users.

References

Hua Ai, Joel R. Tetreault, and Diane J. Litman. 2007. Comparing user simulation models for dialog strategy learning. In *NAACL-HLT*, April.

Carryl L. Baldwin and David Struckman-Johnson. 2002. Impact of speech presentation level on cognitive task performance: implications for auditory display design. *Ergonomics*, 45(1):62–74.

Carryl L. Baldwin. 2001. Impact of age-related hearing impairment on cognitive task performance: evidence for improving existing methodologies. In *Human Factors and Ergonomics Society Annual Meeting; Aging*, pages 245–249.

Linda Bell, Joakim Gustafson, and Mattias Heldner. 2003. Prosodic adaptation in humancomputer interaction. In *Proceedings of ICPhS 03*, volume 1, pages 833–836.

Peter Heeman. 2007. Combining reinforcement learning with information-state update rules. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 268–275, Rochester, NY, April.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Comput. Linguist.*, 34(4):487–511.

J. C. Junqua. 1993. The lombard reflex and its role on human listeners and automatic speech recognizers. *The Journal of the Acoustical Society of America*, 93(1):510–524, January.

Dan Jurafsky, Liz Shriberg, and Debra Biasca. 1997. Switchboard: SWBD-DAMSL Coders Manual.

Blade Kotelly. 2003. *The Art and Business of Speech Recognition*. Addison-Wesley, January.

E. Levin, R. Pieraccini, and W. Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23.

Bjorn Lindblom, 1990. *Explaining phonetic variation: A sketch of the H & H theory*, pages 403–439. Kluwer Academic Publishers.

Rebecca Lunsford, Sharon Oviatt, and Alexander M. Arthur. 2006. Toward open-microphone engagement for multiparty interactions. In *Proceedings of the 8th International Conference on Multimodal Interfaces*, pages 273–280, New York, NY, USA. ACM.

Eric Martinson and Derek Brock. 2007. Improving human-robot interaction through adaptation to the auditory scene. In *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 113–120, New York, NY, USA. ACM.

K. L. Payton, R. M. Uchanski, and L. D. Braida. 1994. Intelligibility of conversational and clear speech in noise and reverberation for listeners with normal and impaired hearing. *The Journal of the Acoustical Society of America*, 95(3):1581–1592, March.

Harvey Sacks, Emanuel A. Schlegoff, and Gail Jefferson. 1974. A simplest sytsematic for the organization of turn-taking for conversation. *Language*, 50(4):696–735, December.

K. Scheffler and S. J. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of Human Language Technology*, pages 12–18, San Diego CA.

A. Stent, M. Huffman, and S. Brennan. 2008. Adapting speaking after evidence of misrecognition: Local and global hyperarticulation. *Speech Communication*, 50(3):163–178, March.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*.

Jessica Villing, Cecilia Holtelius, Staffan Larsson, Anders Lindström, Alexander Seward, and Nina Aaberg. 2008. Interruption, resumption and domain switching in in-vehicle dialogue. In *GoTAL '08: Proceedings of the 6th international conference on Advances in Natural Language Processing*, pages 488–499, Berlin, Heidelberg. Springer-Verlag.

Marilyn A. Walker. 2000. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Aritificial Intelligence Research*, 12:387–416.