# An Adaptive Mediating Agent for Teleconferences

**Rahul Rajan**
Electrical & Computer Engineering
Carnegie Mellon University
rahulraj@cmu.edu

**Ted Selker**
CITRIS
University of California, Berkeley
ted.selker@gmail.com

### Abstract

Conference calls represent a natural but limited communication channel between people. Lack of visual contact and limited bandwidth impoverish social cues people typically use to moderate their behavior. This paper presents a system capable of providing timely aural feedback enabling meeting participants to check themselves. The system is able to sense and recognize problems, reason about them, and make decisions on how and when to provide feedback based on an interaction policy. While a hand-crafted policy based on expert insight can be used, it is non-optimal and can be brittle. Instead, we use reinforcement learning to build a system that can adapt to users by interacting with them. To evaluate the system, we first conduct a user study and demonstrate its utility in getting meeting participants to contribute more equally. We then validate the adaptive feedback policy by demonstrating the agent's ability to adapt its action choices to different types of users.

## Introduction

The Turing test has long captured the imagination of the AI community. The value of a computer mimicking a human, though, should be tied to its relevance in solving a problem. This is particularly important when the AI has to competently work with people in natural settings. To this end we chose the problem of improving the natural but constrained audio communications with AI. We see creating an AI that aids people in working with each other as a potentially more exciting goal than an AI that simply interacts with a person. Adding an agent that can improve the communication between people in such a scenario would demonstrate a productive and useful AI that help people do better on collaborative tasks.

Distributed teams collaborate by holding meetings on conference calls and other networking solutions. By the very nature of the distributed setting, a host of technical, organizational and social challenges are introduced into these meetings that have been well documented, like dominant participants and loud extraneous noises (Yankelovich et al. 2004).



Providing feedback to participants helps them modify their behaviors and address some of the problems that occur in distributed meetings (Erickson and Kellogg 2000; Kim et al. 2008; Yankelovich et al. 2004). In particular, a system that analyzed participant contribution and provided feedback on a separate visual interface, succeeded in getting the participants to contribute more equally to the meeting (Kim et al. 2008).

We explore an AI agent that provides feedback on the same audio channel used by participants in teleconferences. The agent has to be able to interject the communication channel, and provide timely feedback using speech or other audio signals, i.e. aural feedback. This approach opens up a number of issues on how and when to give feedback. While handcrafted interaction policies can be used, it is not possible to design a policy for every situation that might arise. Also, not all users will respond to feedback the same way. To circumvent these issues, we model the agent's interaction with the user as a Markov decision process and investigate if an agent can adapt its behavior to different users using reinforcement learning techniques.

## Audio Conference System

To be capable of facilitating a conference call, the agent analyzes the audio streams from the participants of the meeting, and senses if they are speaking or not, how loudly they are speaking, and whether there is noise on the channel. It then analyzes the interaction between the participants of the meeting, and recognizes social problems when they occur. This is done by computing non-verbal social activity metrics like turn-taking and interruptions, etc. Finally, the system makes decisions on when and how to provide aural feedback to mediate the conference call (Rajan, Chen, and Selker 2012).

### Reasoning Architecture

The research system uses a blackboard architectural model to prioritize and schedule how it responds in a meeting. It consists of multiple Channelizer and one Globalizer blackboards. A Channelizer represents a participant, while the Globalizer represents the meeting.

**Channelizer** Each channelizer classifies microphone input audio as speech or non-speech. The audio is sampled at 16kHz each, with 64 samples per frame. A frame admission process is employed using root-mean-square (RMS) threshold to ignore low-volume events; unless they contain high information, which is determined using a spectral entropy threshold. High information content (like speech) has a lower spectral entropy than a frame with low information (like noise). Spectral entropy is calculated by (i) taking the Fast Fourier Transform (FFT) of a frame; (ii) normalizing it, so as to treat it like a probability mass function (PMF); (iii) and, obtaining the spectral entropy: $H_f = -\sum_{i=1}^{n} p_i \log p_i$.

Admitted frames are processed to extract Mel Frequency Cepstrum Coefficients (MFCC), features that are normally used in speech recognition systems. The MFCC features from a frame are pushed into a sliding window that is 30 frames long. The window is classified into speech and non-speech using a Gaussian Mixture Model (GMM) classifier.

**Globalizer** The Channelizers feeds into the Globalizer which is where the agent makes decisions on when and how to provide feedback to the participants. The Globalizer currently includes three knowledge sources (KSs).

The first KS aggregates audio cues that are non-verbal and have proven to be effective in distinguishing social activity during a meeting (Jayagopi 2011). These include: Total Speaking Length (TSL); Total Speaking Turns (TST); Total Speaking Turns without Short Utterances (TSTwSU); Total Successful Interruptions (TSI). A combination of these yielded an 88% accuracy for classifying conversational dominance on a meeting corpus (Jayagopi 2011). We use the above metrics in our approach.

The second KS determines each speaker's dominance by calculating how active each person is relative to other participants. The Globalizer calculates each participant's dominance as their contribution to the conversation in terms of speaking length (TSL).

The third KS detects and resolves conversational collisions, or interruptions. In collaborative problem-solving meetings, for example, if the agent detects that a person with high dominance is interrupting a person with low dominance, it will set a flag indicating the need to give feedback to the person with high dominance.

The Globalizer maintains an internal queue of problems recognized by the KSs (e.g. dominance, loud noise). It reorders or delays messages based on their importance, the time since the last alert and the number of alerts. It combines similar messages that occur consecutively. Based on its interaction policy, it decides whether, when and how to prompt the user with feedback.

## Adaptive Feedback Policy

Once the agent recognizes the existence of a social problem it attempts to provide feedback based on its interaction policy. The feedback can be parametrized in a number of ways, including its timing, frequency, tone, volume, translucence (Erickson and Kellogg 2000), etc. Hand-crafted feedback policies can be designed based on psychological insights (Rajan, Chen, and Selker 2012). These are often brittle — different users might react differently, and even an individual user's response might change over time and depending on the situation.

**Learning Algorithm** We consider reinforcement Learning as approach to improve social feedback policies. The agent will consider the meeting state based on duration of the meeting, detected social problems, timing and nature of previous feedback, and user's mood. It will also consider feedback actions available to the agent including what type of feedback to give, if any. An agent's action yields some reward $r \in R(s,a)$ and leads to a new state $s' \in S$. In some cases, the desired state in meetings (e.g. non-dominant participants) might occur as a result of several interactions. Such interaction with delayed rewards are well modeled as Markov Decision Processes (MDP).

Solving a Markov process, however, requires knowledge of the possible state transition probabilities (interaction model), which is not known in advance. One way to approach the problem is to use a model-free class of algorithms known as *temporal difference* methods. In particular, we use the Q-learning algorithm (Watkins and Dayan 1992) which is typically easier to implement, where we define $Q^*(s,a)$ as the expected discounted reinforcement for taking action a in state s, then continuing by choosing actions optimally. The Q-learning rule is

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)),$$

where $\alpha$ is the learning rate, and $\gamma$ is the discount factor. $< s,a,r,s' >$ is an experience tuple, as described above. If each action is executed in each state an infinite number of times on an infinite run and $\alpha$ is decayed appropriately, the $Q$ values will converge with probability 1 to $Q^*$ (Watkins and Dayan 1992). The optimal policy then becomes $\pi^*(s) = \arg\max_a Q^*(s,a)$.

**Payoff Function** We focus on **three binary state features**, which are (i) is the participant dominant?, (ii) have they received feedback?, (iii) are they annoyed?. The agent has a choice of **three actions** to get a dominant user to reduce their dominant behavior: *No Action, Advisory, Assistive*. The agent might provide aural *advisory* feedback to the user that they are being dominant. Alternatively the agent might take an *assistive* action: reducing the volume of a dominant person, or muting them when they interrupt a less dominant participant. For meeting flow, it is preferred that the agent chooses (a) no action unless necessary, and (b) advisory over assistive actions. We therefore give the assistive and advisory actions a cost of -5 and -1, respectively. If the user gets annoyed with consecutive feedback actions, the agent incurs a cost of -10. If the agent is able to get a dominant user to change their behavior, without annoying them, it gets a **reward of +50**.

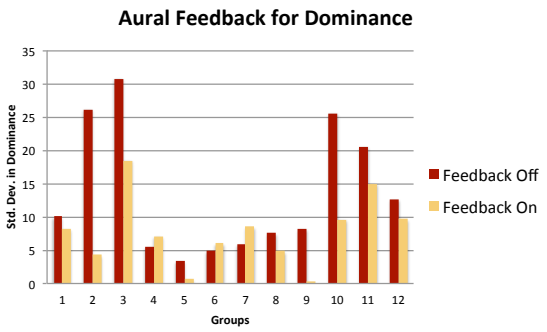



Figure 1: The results of the aural feedback experiment across twelve groups. They demonstrate a reduction in standard deviation of dominance among members of a group, when aural feedback was provided.

## Evaluations

### Aural Feedback

We evaluated aural feedback by having the agent reduce the variance in dominance among participants. Higher variations in dominance between team members leads to less constructive and more passive/defensive interaction styles within teams (Balthazard, Potter, and Warren 2004). During turn-taking conflicts, the agent uses an advisory approach to remind the dominant participant to share the floor by saying "turn-taking?" on that user's channel. Similarly if someone is being dormant, the agent will say "any thoughts?" to encourage their participation.

**Results** 12 groups of 3 participants remotely collaborated for 5 minutes to solve hangman puzzles. With the agent facilitating, the standard deviation[1] in dominance among members of a group reduced with statistical significance (N=12, $p<0.01$, 1-tailed t-test, Figure 1).

### Adaptive Feedback Policy

The Q-learning algorithm was validated for adapting an agent's feedback policy for different users by conducting a set of experiments with a simulated user and environment. In the experiment, an episode starts in any state where the user is dominant. The episode ends when the user is in the goal state, i.e. they are not dominant or annoyed. Thus, there is a trade-off when providing feedback between getting the user to be non-dominant and making sure not to annoy the user. The simulated episodes demonstrate feasibility of the approach; experiments with real users would be a valuable next step for validating and possibly improving the feedback policy.

[1]In a three-person meeting, the ideal contribution is 33.3%, which is also always the average. The standard deviation gives a measure of how close to ideal participant contributions are in each condition. A standard deviation of 0 implies that all the participants contributed equally.

**User Model** A model of potential users focused on their responses to the agent: how did they respond to advisory actions ($R_{Ad}$), how likely they are to get annoyed ($U_{an}$), and how well are they able to self-regulate their behavior without feedback ($U_{sr}$). We would expect that the optimal policy is to do No Action when the user is not dominant or when they are annoyed. This was the case in all the optimal policies that were learnt. Thus, we are left with two states, i.e., (i) $S_{D\bar{F}}$: user is dominant and has not gotten any feedback, and (ii) $S_{DF}$: user is dominant and has received feedback, where the agent learns different policies.

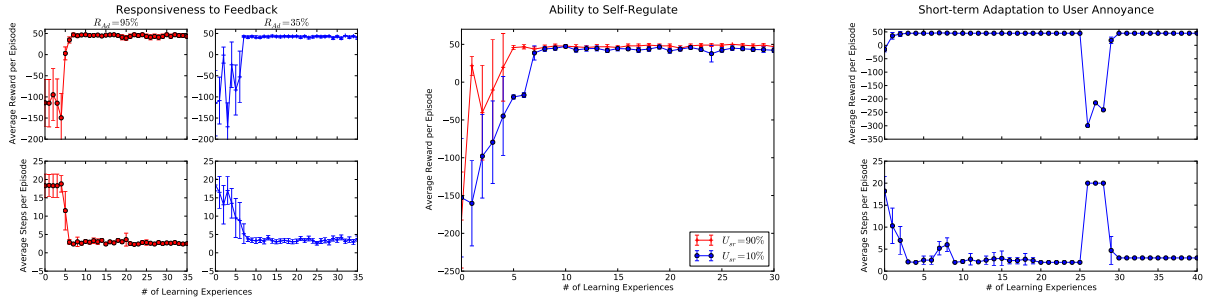
**Results** For the following experiments, an agent is trained using a series of *learning experiences*. A learning experience can consist of one or a batch of episodes. During the learning experience the agent uses an $\epsilon$-greedy explorer to choose an action. An $\epsilon$-greedy explorer chooses a random action with $\epsilon$ probability, and choose an action using the learnt policy with 1-$\epsilon$ probability. After each learning experience, the new policy is tested over 10 test episodes. During these, the agent always chooses an action based on the learnt policy. The rewards the agent receives over the 10 test episodes is averaged and plotted in Figure 2.

*Responsiveness to Feedback*
Two types of users were simulated with different responsiveness to advisory feedback, i.e. the probability with which a dominant user will become non-dominant when they get advisory feedback: $R_{Ad} = \{95\%, 35\%\}$. The user always responds to assistive feedback with a probability of 95%. If the user has been provided feedback, the likelihood of them responding to advisory feedback drops by 10% in the next attempt. The optimal policy that was learnt was to provide advisory actions in $S_{D\bar{F}}$ and $S_{DF}$ when $R_{Ad} = 95\%$, i.e. the agent learns that the user is likely to respond to advisory feedback. When $R_{Ad} = 35\%$, the agent chose assistive actions in both states, because it learns that the user is unlikely to respond to advisory feedback, and that it has to pursue the less desirable (more costly) assistive actions.

Figure 2a plots the rewards and the number of actions the agent took as it learnt an optimal policy for $R_{Ad} = 95\%$ & $R_{Ad} = 35\%$. These results are averaged over 10 test episodes after every learning experience.

*Ability to Self-Regulate*
Next, we model a user who is dominant only for short periods of time. In this case, we include a likelihood that the user becomes non-dominant when the agent takes no action ($U_{sr}$) to the existing ($R_{Ad} = 35\%$ + $U_{an}$) user model. The agent was trained for two cases: $U_{sr} = \{10\%, 90\%\}$. When $U_{sr} = 10\%$, the agent learns the same policy as the $R_{Ad} = 35\%$ model, since the user does not self-regulate and needs to receive assistive feedback to become non-dominant. When $U_{sr} = 90\%$, the agent chooses to do no action in every state because it learns that the user is likely to self-regulate, and does not need feedback.

Figure 2b plots the rewards earned as the agent learns an optimal policy for $U_{sr} = 90\%$ and $U_{sr} = 10\%$. The higher rewards for $U_{sr} = 90\%$ are indicative of the agent choosing




(a) (b) (c)

Figure 2: The results of the adaptive feedback policy experiments. Each figure shows results with (95% confidence interval) error bars averaged over 10 test episodes, for every learning experience. In all cases, the agent learns an optimal policy in under 10 experiences.

no action (no cost), while the lower rewards for $U_{sr} = 10\%$ indicate the agent choosing assistive actions (-5 cost).

*Short-term Adaptation to User Annoyance*
In this experiment, we also test the agents short-term adaptation to new information once it has already learnt an optimal policy. We add to an existing user model ($R_{Ad} = 35\%$) the likelihood of them getting annoyed with consecutive feedback ($U_{an}$). In state $S_{DF}$, the agent should learn to take no action instead of providing assistive feedback.

Figure 2c plots the results as the agent learns an optimal policy for $R_{Ad} = 35\%$. After the 25th learning experience, the user begins to get annoyed with consecutive feedback ($U_{an}$). The plot shows how the agent is punished for following the optimal policy when this happens, and how it adapts after 3 to 4 learning experiences to learn a new optimal policy for $\{R_{Ad} = 35\%, U_{an}\}$.

## Conclusion & Future Work

This work validates a methodology for creating an AI that can coexist in an audio medium to improve meeting interactions between participants. The system's use of two layers of blackboards allow it to judge the relative contributions of collaborators. An agent that decided when to give advisory support to improve collaboration has been created and validated. Future work should compare our simulated users to a corpus from real users for improving the adaptive feedback policy. The work could also be extended by implementing a larger state space that can take into account other factors like the length of the meeting.

An AI agent might improve a conversation in several ways (Rajan, Chen, and Selker 2012; Rajan et al. 2013). We are excited about successes of using AI to participate in audio meetings to improve peoples relative contributions. Of course not all meetings are collaborations, they might be dissemination meetings, or round-robin status update meetings, etc. This paper could help point the way to models that evaluate the meeting as well as the user. These could help inform the agent's interaction policy based on the type of meeting, state of the meeting, as well as the state of the user. This work might encourage others to explore new ways of creating AI that give social feedback in user interfaces to improve human performance.

## References

Balthazard, P.; Potter, R. E.; and Warren, J. 2004. Expertise, Extraversion and Group Interaction Styles as Performance Indicators in Virtual Teams: How do Perceptions of IT's Performance get formed? *J. SIGMIS Database* 35(1):41–64.

Erickson, T., and Kellogg, W. A. 2000. Social Translucence: An Approach to Designing Systems that Support Social Processes. *Proc. CHI 2000* 7(1):59–83.

Jayagopi, D. B. 2011. *Computational Modeling of Face-to-Face Social Interaction using Nonverbal Behavioral Cues*. Ph.D. Dissertation, Ecole Polytechnique Fédérale de Lausanne.

Kim, T.; Chang, A.; Holland, L.; and Pentland, A. S. 2008. Meeting Mediator: Enhancing Group Collaboration using Sociometric Feedback. In *Proc. CSCW 2008*, CSCW '08, 457–466.

Rajan, R.; Hsiao, J.; Lahoti, D.; and Selker, T. 2013. roger that!the value of adding social feedback in audio-mediated communications. In *Human-Computer Interaction–INTERACT 2013*. Springer. 471–488.

Rajan, R.; Chen, C.; and Selker, T. 2012. Considerate Audio MEdiating Oracle (CAMEO): Improving Human-to-Human Communications in Conference Calls. In *Proc. DIS 2012*, DIS '12, 86–95. ACM.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3):279–292.

Yankelovich, N.; Walker, W.; Roberts, P.; Wessler, M.; Kaplan, J.; and Provino, J. 2004. Meeting Central: Making Distributed Meetings more Effective. In *Proc. CSCW 2004*, CSCW '04, 419–428.