An Adaptive Mediating Agent for Teleconferences

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

ABSTRACT

Conference calls are an important tool for today's global businesses. The use of this technology is however plagued by a number of social problems that impact meeting effectiveness, which can be attributed to the missing visual channel and low bandwidth for non-verbal signals that people 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 actions choices to different types of users.

1. INTRODUCTION

Globalization and technology have brought radical changes to business practices, and meetings in particular. With increasing frequency, teams are being composed of members from geographically different locations so that they can bring to bear their expertise on pressing problems, without the travel and associated costs. These 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 [8].

It has been shown that providing feedback to participants helps them modify their behaviors and address some of the problems that occur in distributed meetings [2, 4, 8]. Feedback can be about a participant expressing their desire to talk, low microphone volume, etc., and is usually provided on an alternate channel (or medium) that runs parallel to the meeting, without disrupting it. In particular, a system that analyzed participant contribution and provided automated visual feedback, succeeded in getting the participants to contribute more equally to the meeting [4].

In this work, we explore the idea of a system providing feedback to meeting participants, specifically for audio teleconferences. This introduces the constraint that the agent has to share the same audio channel that is being used for the meeting. 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 hand-crafted 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.

This paper is organized as follows: after a survey of related work, we describe the design and architecture of a testbed system that facilitates audio conference calls between two or more people. We then describe the modeling of the interaction between the user and the agent so as to allow the agent to learn an optimal feedback policy for each user. Next, we present a series of evaluations: (i) the system is evaluated on real users using a hand-crafted policy to validate the utility of aural feedback. We demonstrate the resolution of conversational dominance, a common social problems in meetings. (ii) we validate the agent's capacity to adapt to different users by modeling their responsiveness to feedback, irritability, and ability to self-moderate their own behavior. We conclude by identifying directions for future work.

2. RELATED WORK

Many researchers have tried to overcome the shortcomings of distributed collaboration. Erickson and Kellogg formulated the concept of social translucence [2] to facilitate fluid and productive online group interactions. Their ideas were employed by Yankelovich, et al., in the design of the Meeting Central system to address the problems with audio conferencing which were documented in a series of studies [8]. The problems were grouped into three categories: *audio, behavioral, and technical*. More interestingly, the authors note that "most audio problems are, in fact, behavioral. They are compounded by the difficulty remote participants have, both technically and socially, in interrupting to indicate that the problem exist" [8].

The idea of using feedback to influence group dynamics and behavior in distributed meetings was further explored by Kim et al. in Meeting Mediator [4]. They focussed primarily on the effects of feedback on dominant meeting participants. Their Meeting Mediator system computes group interactivity and speaker participation levels, and uses a visualization to provide feedback to the participants on their personal mobile devices.

The work discussed so far uses GUIs and visualizations of a reactive and peripheral nature. The question arises as to how facilitation can be done when there are no displays available like on a telephone conference call, or when the display is being used to view shared artifacts. Also, since spoken communications are so dynamic how can facilitation be achieved at the turn-taking level, to manage interrupts or overlaps for example. One solution is to build proactive agents that provide timely aural feedback. In the next section, we describe the design and architecture of the testbed system we built to facilitate audio conference calls.

3. AUDIO CONFERENCE SYSTEM

To be capable of facilitating a conference call, the system needs to analyze the separate audio streams from the participants of the meeting and *sense* if they are speaking or not, how loudly they are speaking, and whether there is noise on the channel. It then has to *analyze* the interaction between the participants of the meeting, and *recognize* social problems when they occur. This is done by computing non-verbal social activity metrics like turn-taking and interruptions, etc. Finally, the system needs to make decisions on when and how to provide aural *feedback* to mediate the conference call.

3.1 Reasoning Architecture

In its current implementation, the system uses a blackboard architectural model to prioritize and schedule how it responds in a meeting. It consists of Channelizer and Globalizer blackboards. The Channelizer represents a participant and hence is local in its scope. The issues of an individual participant are resolved here. The Globalizer represents the meeting. It interprets the different dynamics of the meeting and organizes the agent's feedback actions.

3.1.1 Channelizer

The system maintains a Channelizer blackboard for each participant, which consist of Knowledge Sources (KS) that keep track of participant activity:

The first KS 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. A frame with high information content (like speech) has a lower spectral entropy than a frame with low information frame (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, by

$$H_f = -\sum_{i=1}^{n} p_i \log p_i$$

Admitted frames are further 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 step size is of one buffer, i.e. there is no overlap. The window is then classified into speech and non-speech using a Gaussian Mixture Model (GMM) classifier trained using the Expectation-Maximization (EM) algorithm.

The second KS calculates signal-to-noise ratios which are used to detect extraneous noise, and to determine if a participant is speaking too loudly or softly. Speech buffers are used to determine signal values, while non-speech buffers are used to determine noise floor values.

3.1.2 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.

The first KS aggregates several audio cues that are non-verbal, and have proven to be effective in distinguishing a speaker's social activity during a meeting [3]. These include: Total Speaking Length (TSL); Total Speaking Turns (TST); Total Speaking Turns without Short Utterances (TSTwSU); Total Successful Interruptions (TSI). Jayagopi showed that using a combination of these cues to classify conversational dominance yielded an 88% accuracy on a fairly typical meeting corpus [3], which is why we chose the above metrics in our evaluation process.

The second KS determines each speaker's dominance by calculating how active each person is relative to the activity level of the other participants. The Globalizer calculates each participant's dominance as their contribution to the sum of all participants, where TSL is the Total Speaking Length of a particular participant:

$$Dominance(P_x) = \frac{TSL(P_x)}{\sum_{i=1}^{n} TSL(P_i)}$$

The third KS detects and resolves any conversational collisions, or interruptions. In collaborative problem-solving meetings, for example, if the agent detects that a participant with high dominance is interrupting a participant with low dominance, it will set a flag indicating the need for the agent to provide feedback to the participant 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. Finally, based on its interaction policy, the agent decides whether, when and how to prompt the user with feedback to address the problems.

3.2 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 [2], etc. Hand-crafted feedback policies can be designed based on psychological insights, but these are often non-optimal and brittle — different users might react differently to these parameters, and even an individual user's response will change over time depending on the situation. The feedback needs to be personalized to the user, and must be able to adapt to the situation. A similar problem for cognitive orthotics was addressed using reinforcement learning techniques [6].

3.2.1 Learning Algorithm

Reinforcement Learning is an approach to induce optimal feedback policies ($\pi^* : S \rightarrow A$) as a function of the current state $s \in S$ (e.g. duration of the meeting, detected social problems, timing and nature of previous feedback, user's mood) and feedback actions available to the agent $a \in A$, 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 an MDP, however, requires knowledge of all the possible state transition probabilities (interaction model), which is not known in advance for each meeting/user. 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 [7] 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 α is the learning rate, and γ 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 α is decayed appropriately, the Q values will converge with probability 1 to Q^* [7]. The optimal policy then becomes $\pi^*(s) = \arg\max_a Q^*(s,a)$.

3.2.2 Payoff Function

In this work, we focus on **three binary state features**, which are (i) is the participant dominant?, (ii) have they received feedback?, (iii) are they annoyed?. This gives a total of 8 states that the user can be in. The agent has a choice of **three actions** to get a dominant user to reduce their dominant behavior: *No Action, Advisory, Assistive.* Advisory is when the agent provides aural feedback to the user that they are being dominant. Assistive is when the agent autonomously reduces the volume of a dominant person, or mutes them when they interrupt a less dominant participant. It is preferred that the agent chooses (a) no action unless necessary, and (b) advisory over assistive actions. For this reason, the assistive and advisory actions have an associated cost of -5 and -1, respectively. Also, if the user gets annoyed (with consecutive feedback actions), the agent incurs a cost of -10. Finally, if the agent is able to get a dominant user to change their behavior, without annoying them, it gets a **reward of +50**.

4. EVALUATIONS

This section presents a series of evaluations. Firstly, the audio conference system is evaluated to demonstrate the utility of aural feedback in a real meeting [5]. Secondly, we demonstrate how the Q-learning algorithm can be used to learn optimal feedback policies that adapt to different users.

4.1 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 [1]. To encourage more extraversion, the agent keeps track of participant contributions. During turn-taking conflicts, the agent uses its advisory approach to remind the dominant participant to share the floor by saying "turn-taking?" on that users channel. Similarly if someone is being dormant, the agent will say "any thoughts?" to encourage their participation.

4.1.1 Results

Twelve groups of three participants were tasked with remotely collaborating for 5 minutes to solve hangman puzzles. We showed that with the agent facilitating, the standard deviation in dominance among members of a group reduced with statistical significance (Standard deviation in Dominance[1], N=12, p<0.01, 1-tailed t-test, Figure 1).

[1]In a three-person meeting, the ideal contribution is 33.3%, which is also always the average. The standard deviation gives us 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.

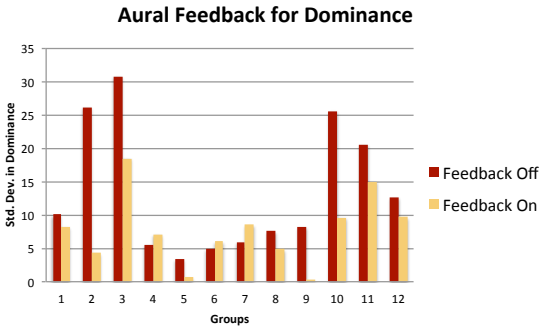


Figure 1: The results of the aural feedback experiment across twelve groups. They demonstrate a reduction in standard deviation of dominance among members of a group, when aural feedback was provided.

4.2 Adaptive Feedback Policy

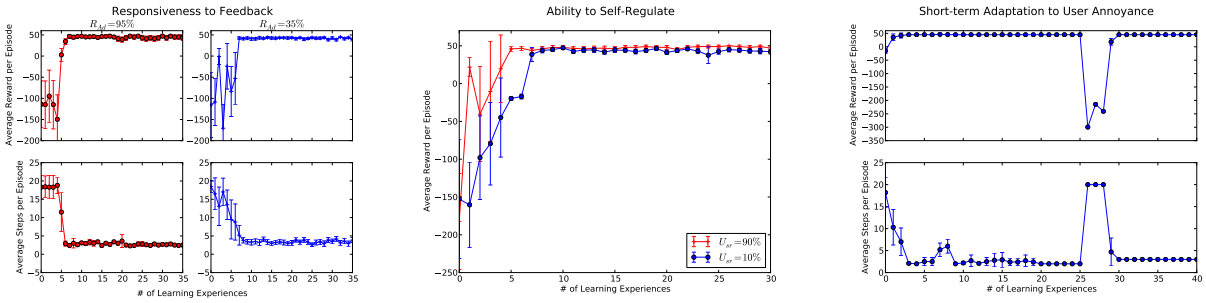
The applicability of the Q-learning algorithm to enable an agent to adapt its feedback policy and learn an optimal policy for different users was validated by conducting a set of experiments with a simulated user and environment. The experiments are conducted *episodically.* To start an episode, the environment is initialized to a random 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 tradeoff when providing feedback between getting the user to be non-dominant and making sure not to annoy the user. Reliance on simulated users is highly imperfect, and to truly validate our results we will need to run experiments with real users. However, it was necessary to first demonstrate the feasibility of our approach in principle.

4.2.1 User Model

We attempted to build a realistic model of potential users by focussing on three key relevant aspects of their behavior: how they respond to advisory actions (R_{Ad}), how likely they are to get annoyed (U_{an}), and how well they are able to self-regulate their behavior without feedback (U_{sr}). Regardless of any of these aspects we would expect that the optimal policy is that the agent does 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.

4.2.2 Results

For the following experiments, an agent is trained using a series of *learning experiences.* A learning experience can consist of an episode or a batch of episodes. During the learning experience the agent uses an ϵ-greedy explorer to choose an action. An ϵ-greedy explorer choses a random action with ϵ probability, and choose an action using the learnt policy with 1-ϵ probability. After each learning experience, the newly learnt policy is tested over 10 test episodes. During the test episodes, 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.



(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 learning experiences.

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%. Furthermore, 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, i.e. they can regulate their behavior without feedback. 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 choses 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 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 $\{R_{Ad} = 35\%, U_{an}\}$.

5. CONCLUSION & FUTURE WORK

This work addresses the social problems that occur on conference calls through a mediating agent that provides aural feedback. This approach was evaluated in a user study and shown to be effective in reducing dominance with statistical significance. Since the communication channel is shared between the participants and the system, however, a different set of problems has to be addressed, as to how and when feedback should be provided. An adaptive interaction policy was implemented using reinforcement learning techniques, and validated using a simulated user and meeting environment. In future work, we will implement a larger state space for reinforcement learning that can take into account other factors like the length of the meeting, and its different phases. Lastly, we would want to evaluate the adaptive feedback policy on users in real meetings.

6. REFERENCES

[1] P. Balthazard, R. E. Potter, and J. Warren. 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, Feb. 2004.

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

[3] D. B. Jayagopi. *Computational Modeling of Face-to-Face Social Interaction using Nonverbal Behavioral Cues*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2011.

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

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

[6] M. Rudary, S. Singh, and M. E. Pollack. Adaptive Cognitive Orthotics: Combining Reinforcement Learning and Constraint-Based Temporal Reasoning. In *Proc. ICML*, page 91. ACM, 2004.

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

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